Secure Model Management Operations for the Web

Guanglei Song, Kang Zhang, Bhavani Thuraisingham, and Jun Kong

University of Texas at Dallas, Richardson, Texas 75083-0688 USA
{gxs017800, kzhang, bhavani.thuraisingham, jxk019200}@utdallas.edu

Abstract. The interoperability among different data formats over the Internet has drawn increasing interest recently due to more and more heterogeneous data models are used in different Web services. In order to ease the manipulation of data models for heterogeneous data, generic model management has been intensively researched and also implemented in a prototype since its first introduction. Access control specifications attached to each individual data model require significant amount of efforts to manually specify. Based on a general security model for access control specifications on heterogeneous data models and its visual representation, we present secure model management operators for managing access control specifications.The secure model management operators disccussed in the paper include a secure match operator and a secure merge operator. We introduce a novel graphical schema matching algorithm and extend the algorithm to make a secure match operator. The paper also discusses secure merge principles for the integration of data models.

1 Introduction

The huge success of the Web as a platform for information dissemination has brought an increasing awareness of the fact that document exchange over the Internet should meet security requirements such as fine-grained authenticity and access control [24]. XML [4,5] and database [9] access control models have been a hot research topic. Recently, the continuing demand for information sharing has shifted interests from stand-alone XML repositories and databases to interconnected and large-scale cooperative systems [6].

Manually manipulating heterogeneous data models has been a time-consuming and error-prone process. Therefore a new approach to metadata management, i.e. Model Management, has been proposed [2]. Model management offers a high-level programming interface and avoids object-at-a-time primitives by manipulating models with generic operators. Our previous work provides a visual model management architecture, which eases the use of the generic operators [23]. The visual architecture, however, does not provide secure interfaces for managing access control specifications, which are associated with data models. These specifications can only be managed manually, and the procedure for managing secure models, therefore, cannot be fully realized by current model

S. Jajodia and D. Wijesekera (Eds.): Data and Applications Security 2005, LNCS 3654, pp. 240–254, 2005.

management systems. This paper focuses on the security properties of model management, and explores various issues and solutions to achieve secure model management of data models.

One challenge of secure model management comes from the heterogeneity of data formats. Data encoded in different formats needs to be exchanged in cooperative systems, thus achieving interoperability. Even though every individual data model may have highly secure access control specifications and enforcement mechanism, the federation of data models is not necessarily secure. Security of a union of systems is determined by the weakest link. When information of different models is interchanging, it opens a window for attack. The security extensions presented in the paper ease the manipulation of models with access control specifications and provide a guidance for generating safe mappings and unions of models.

The remainder of the paper is organized as follows. Section 2 introduces a uniform access control model and an illustrative example. Section 3 proposes a graphical schema matching algorithm and the security extension to the algorithm. Section 4 presents security extensions to other model management operators. Section 5 discusses the future research directions. Section 6 compares related works and Section 7 concludes the paper.

2 A Uniform Access Control Model

Our uniform access control model consists of a set of rules, each being a tuple of five elements: *subject*, *object*, *action*, *authorization*, and *propagation* [24]. Access control regulates access to the data, such as XML documents, and databases called *objects*. Those who try to access these objects are called *subjects*. A subject is represented by a unique user-defined identifier called UPath [24], e.g. tables and columns of a relational schema, elements and attributes of an XML schema. Actions include *read*, *write*, *update*, and *delete*. An authorization specifies the negative or positive response to a request, i.e. *allow* or *deny*. The propagation can be either *local* or *recursive*, referring to the influence of the object locally or recursively to its child objects.

We visualize the access control rules by node-edge diagrams [24]. As shown in Figure 1, a rule is represented as a link and a subject is represented by a labeled rectangle connecting to objects (as labeled ellipses). A gray eclipse represents recursive access, and a white eclipse indicates local access. The label of each link, R or W, represents the activity. The circle and the cross on a link represent *allow* and *deny* of access respectively.

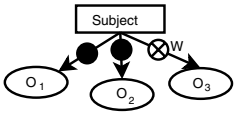

Fig. 1. Visual representation of access control

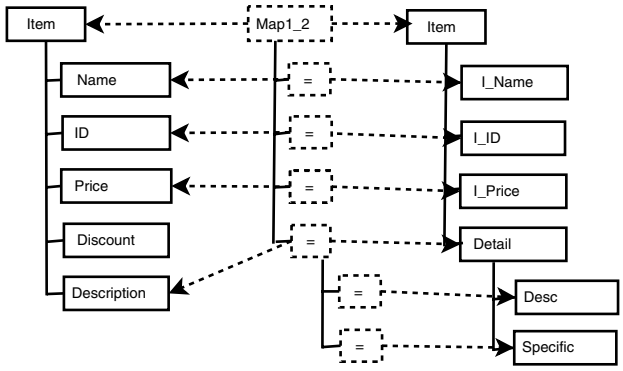

Fig. 2. Online shopping schemas for two companies

Table 1. Access Rules for Model A

	Subject	Object	Action	Authorization	Propagation
1	Customer	/Item/	Read	Allow	R
2	Vendor	/Item/	Read	Allow	R
3	Administrator	/Item/	Read	Allow	R
4	Administrator	/Item/Description	Write	Allow	L
5	Administrator	/Item/Discount	Write	Allow	L
6	Administrator	/Item/Price	Write	Allow	L

Table 2. Access Rules for Model B

	Subject	Object	Action	Authorization	Propagation
1	Cust	/Item/	Read	Allow	R
2	Provider	/Item/	Read	Allow	R
3	Admin	/Item/	Read	Allow	R
4	Admin	/Item/Detail	Write	Allow	R
5	Admin	/Item/Price	Write	Allow	L

Consider the following example. Two companies A_c and B_c want to offer a joint online solution for customers and vendors. Figure 2 shows the two schemas, A and B, for companies A_c and B_c.

Companies A_c and B_c have local access control rules as shown in Tables 1 and 2 respectively.

A model management system eases the process by providing generic operators like Match and Merge. Figure 3 shows the scenario of unifying the two models by the two operators [24]. ACR_A and ACR_B are access control rules for models M_A and M_B respectively. M_u is the unified model of M_A and M_B. ACR_u is a set of access control rules for model M_u. The system matches and merges M_A and M_B to generate M_u, but cannot automatically generate ACR_u. Users

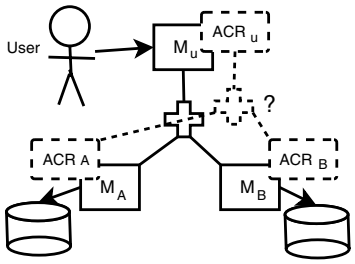

Fig. 3. Unified online shopping system

have to construct ACR_u manually from scratch. It is highly risky to manually manipulate access control rules in a large scale such as an online store site. To ease the process, a security extension for model management operators (like Match) is desirable for automatically managing access control rules.

3 Secure Schema Matching

This section introduces a new matching algorithm for graphical generation of schema mappings, and adapts the algorithm by adding a security extension.

3.1 A Graphical Schema Matching Algorithm

Schema matching is to find semantic correspondences among elements of two schemas. Most of the proposed approaches [20] concentrated on the similarity of individual elements or at most neighborhood information, rather than on the global semantics of the schemas. We propose a novel approach to the schema matching problem utilizing global semantics. A schema is represented by an acyclic directed graph, where nodes represent elements or attributes and links represent the containment relationships.

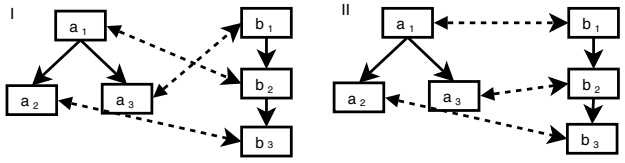

Fig. 4. I.A mapping with semantic contradiction; II. A harmonic mapping

Schemas are represented by acyclic graphs, which do not allow containment cycles that cause a *semantic contradiction*. If a semantic mapping between two schemas has no semantic contradiction, we call the mapping *harmonic*. Figure

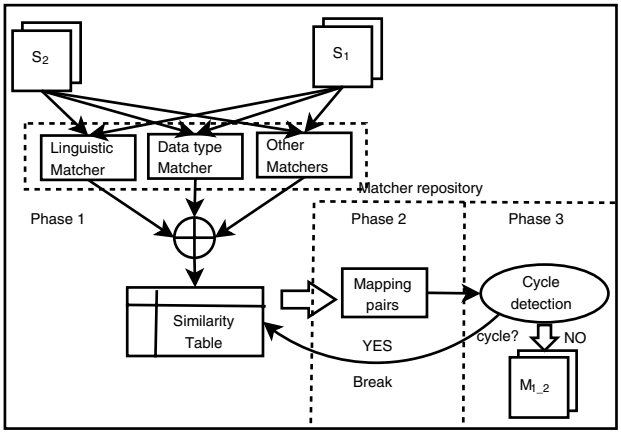

Fig. 5. Mapping generation process in GGS

4 shows an example of mapping with semantic contradiction and the other being harmonic. Figure 4.I includes mapping pairs (a_1, b_2) and (a_3, b_1), while $a_1$contains $a_3$and b_1 contains b_2. From the graphical representation of mapping, the relationships between (a_1, b_2) and (a_3, b_1) produce a non-harmonic crossing, which results in a semantic contradiction when applying the mapping. For example, in case of merging elements of the schemas based on the mapping in Figure 4.I, how can an algorithm decide whether the mapping (a_1, b_2) should contain the mapping (a_3, b_1) or the other way? Also the mapping pairs are self-contradictory due to $a_1 \rightarrow a_3 \leftrightarrow b_1 \rightarrow b_2 \rightarrow a_1$, i.e. a containment cycle, which is contradictory to the acyclic representation of schemas.

A harmonic mapping, such as the one in Figure 4.II, is desirable. We present a schema matching algorithm for producing harmonic mappings. Our schema matching algorithm proceeds in 3 phases as shown in Figure 5:

1. Use various types of matchers to compare element names and calculate similarity of data types and produce a similarity table for each pair of elements;
2. Produce an initial set of mapping pairs by selecting possible mappings from the initial similarity table;
3. Search and break cycles that exist, and go to Phase 1 for the next iteration until no cycle exists or when the number of iterations reaches a predefined upper bound (beyond which the computational cost is no longer worthwhile for users).

A single matcher generates a similarity table consisting of similarity values for any two input elements. A similarity value is a number between 0 (strong dissimilarity) and 1 (strong similarity). Our matching algorithm combines these similarity tables by computing weighted averages. Assume n single similarity tables, $table_1$ to $table_n$, each having a similarity value Sim_i (a, b), i =1..n, for any pair of elements (a, b). For each pair of elements (a, b) from schemas A and B, the overall similarity Sim (a, b) can be calculated by:

$Sim(a,b) = \frac{\sum_{i=1}^{n}(Sim_i(a,b)\times w_i)}{n}$, where $\sum_{i=1}^{n}(w_i) = 1$, and Sim_i is a similarity table produced by matcher i.

Phase 2 generates mapping pairs based on the combined similarity table by choosing the best match for each element in the table. Then in Phase 3, our matching algorithm, as shown in Algorithm 3.1, detects cycles by checking the decedents and ancestors of each element of a mapping pair to see if they contribute a pair in the mapping. If so, a cycle exits.

Algorithm 3.1 cycle detection

Require: Table table
  for each pair p(a_i, b_j) in pairs; **do**
    Topsort(pairs);
    if ((decedents(a_i) X ancestors(b_j)) U (ancestors (a_i) X decedents (b_j))) ∩ pairs ≠ empty) **then**
      returntrue;
    end if
  end for

If a cycle exists in the mapping pairs, the algorithm needs to choose a mapping pair to adjust to remove the cycle. Conceptually, the pair that generates the most crossings should be removed, i.e. the key contradiction pair. For the example in Figure 4, the initial mapping pairs are produced from the similarity table as {(a_1, b_2), (a_2, b_3), (a_3, b_1)}. The algorithm detects that mapping pair (a_1, b_3) produces most contradictions with other mappings pairs, and therefore the pair as the key mapping pair needs to be adjusted. Then our algorithm breaks the cycle by finding the second most suitable mapping for the element in the table. Algorithm 3.2 describes the procedure: it finds the key contradiction by calculating the maximal intersection set between mapping pairs and decedents and ancestors of a mapping pair, and then chooses the second best mapping for the element.

Algorithm 3.2 breaking cycles

Require: Table table
  for each pair p(a_i, b_j) in pairs; **do**
    Topsort(pairs);
    if (|((decedents(a_i) X ancestors(b_j)) U (ancestors (a_i) X decedents (b_j))) ∩ pairs | is max) **then**
      Choose the second biggest similarity for a_i
    end if
  end for

After breaking cycles, the algorithm generates mapping pairs based on the new similarity table and iteratively finds and breaks new crossings until no more

Algorithm 3.3 the matching algorithm

```
Require: Data models m1 and m2
  Table table = construct (m1, m2);
  Itno = 0;
  while into< bound do
    if (cycle (table) then
      Break(table);
    end if
    Into ++;
  end while
  Produce the mapping pairs from table;
```

crossing can be found or a threshold is reached. The pseudo code is described in Algorithm 3.3.

3.2 Schema Matching with Security Property

The Match operator takes models A and B as input, and produces mapping Map_{1_2}, called *object mapping*, but not mapping for access control rules. *Subject matching* matches the subjects of two access control rules. For example, "Cust" in Table 1 is mapped to "Customer" in Table 2. Match with a security extension takes two input models, each having a set of access control rules. The extended Match operator is defined as follows:

Definition 1: $(Map_o, Map_s) = Match\ ((M_1,\ ACR_1), (M_2,\ ACR_2))$, where M_1 and M_2 are two data models, ACR_1 and ACR_2 are access control rules of models M_1 and M_2 respectively.

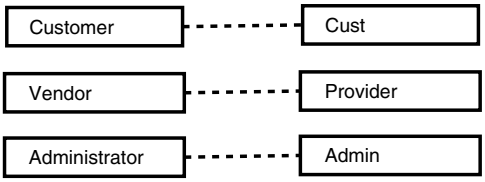

Fig. 6. The Maps example

The result (Map_o,Map_s) contains two mappings, Map_o between objects and Map_s between subjects. Figure 6 shows an example of subject mapping.

Object matching algorithms can be used for subject matching without considering security properties of access control rules, and thus may produce poor and even risky mappings. A security extension of match should avoid risky mappings and produce safe mappings defined as follows.

Assume models M_1 and M_2 have access control rules ACR_1 and ACR_2 respectively. S_1 and S_2 are subjects of ACR1 and ACR_2. Map_{1_2} is the object mapping between M_1 and M_2. Map_s is a subject mapping between S_1 and S_2.

Definition 2: Map_s is *safe* if and only if ($\forall$ (s_1, s_2) $\in Map_s$ $\forall$ (o_1, o_2) $\in Map_{1\_2}$ a $\in$A (allow (s_1, o_1, a) iff allow (s_2, o_2, a))), where s_1 and s_2 are subjects of S_1 and S_2, o_1 and o_2 are objects of M_1 and M_2, a is an action in set A (all actions), allow (s_1, o_1, a) means that s_1 is allowed to perform action a on o_1.

To produce safe subject mappings, three options can be considered:

1. Security filter: The most straightforward approach is to transplant an object matching algorithm to match subjects with a security filter attached to the back end. Once the filter finds a violation, it removes the mapping pair. The approach is safe, but may impair effectiveness of the matching algorithm. For example, as described in Section 2, if another element called Supervisor in model B has full access to model B like Administrator of model A, and security filter cannot match Supervisor with Administrator, since Administrator is chosen to match Admin in the first place.

2. Security dimension: The approach provides security as another dimension of similarity. Careful scrutiny of this approach shows that violating mappings may be produced due to the influence of the other dimensions of similarity (e.g. data type or naming similarity).

3. Security isomorphism: The approach calculates the similarity of subjects based on not only subject names but also semantics of access control rules. It compares the access control rules of every pair of subjects from the graphical representation and generates subject mapping based on the isomorphism of ACRs.

Among the above three options, the security isomorphism algorithm generates more accurate mappings than security filter does, and can also be proved to be safe. Therefore we choose the third approach as the security extension to our matching algorithm presented in Section 3.1. The algorithm matches subjects' access rules to calculate the similarity of two subjects. The similarity of two subjects consists of SS (subject similarity) and AS (access similarity). If s is a subject of access rules, we denote G (s) as a set of objects that S has access and D (s) as the set of objects that S is prohibited from access.

Definition 3: The *overlap set* between two subjects is defined as: O (s_1, s_2) = {(o_1, o_2)| o_1 $\in$G (s_1) and o_2 $\in$ G (s_2), and s_1 and s_2 are two subjects, and (o_1, o_2) is a mapping}.

Definition 4: *The access similarity* between two subject nodes is defined as: $AS(s_1, s_2) = |O(s_1, s_2)|/ N$, where $N = |G(s_1)| + |G(s_2)| - |O(s_1, s_2)|$, and no mapping (o, p) exists such that $o \in G(s_1)$ and $p \in D(s_2)$ or $o \in G(s_2)$ and $p \in D(s_1)$. Otherwise, $AS(s_1, s_2) = -1$.

As shown in Figure 7, the overlap set O (Vendor, Provider) = {(1, a)}. We use Algorithm 3.4 to compute the similarity of two subjects, and then match subjects by choosing the best match in the similarity table.

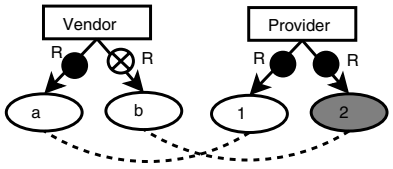

Fig. 7. Matching for access control rules

Algorithm 3.4 Subject matching

Require: Data models o1, o2, subjects s1, s2
Use graphical global schema matching algorithm to produce subject similarity table $SS(s_1, s_2)$ and mapping M;
$AS(s_1,s_2) = 0$;
for each pair of subjects **do**
  for every rule in ACR **do**
    if allow(s_1, o_1, a) and allow(s_2, o_2, a) and (o_1,o_2) $\in$ M **then**
      $O(s_1, s_2) \Downarrow (o_1,o_2)$;
    end if
    if violation exists **then**
      $AS(s_1, s_2)$=-1; break;
    end if
    if $(AS(s_1,s_2)$!= -1 **then**
      $AS(s_1, s_2) = |\ O(s_1, s_2)|/\ N$;
    end if
    if $AS(s_1, s_2)$ >=0 **then**
      $SIM(s_1, s_2) = w^* AS(s_1, s_2) + (1\text{-}w) * SS(s_1, s_2)$;
    else
      $SIM(s_1, s_2)$ = -1;
    end if
  end for
end for
for each subject s_1 in S_1 **do**
  if Max ($SIM(s_1, s_2)$) and $SIM(s_1,s_2)$ >0 **then**
    $ap_s \Downarrow (s_1, s_2)$;
  end if
end for

Theorem 1: Algorithm 3.4 generates safe mappings.

Proof: The algorithm computes the similarity between any pair of subjects in two input models based on the object mapping. Any possible violation will be identified by marking the semantic similarity as -1. The AS value will finally prevent mapping between any two violating subjects. Hence the algorithm generates the mapping between those pairs of subjects that have no possible violation of access control rules. According to Definition 2, the generated mapping is safe.

4 Merge with Security Property

Having the mapping between two models, one can merge the two models to generate a federation and exchange information. The security extension of Merge eases the process by automatically generating access control rules for the output data model. We define the Merge operator with security extension as the following:

Definition 5: $(M_3, ACR_3, Map_{1\_3}, Map_{2\_3}) = Merge$ (M_1, M_2, Map_{1_2}, ACR_1, ACR_2, Map_a), where M_1 and M_2 are input data models, and Map_{1_2} represents the mapping between M_1 and M_2. Map_a represents the mapping between two access control rules $ACR_1$and ACR_2. A Merge operator generates M_3, Map_{1_3},

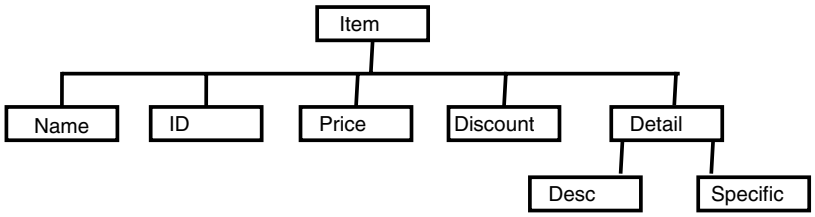

Fig. 8. Result merged schema

and Map_{2_3}. The result model M_3 for the previous example is shown in Figure 8. Mapped elements in M_1 and M_2 are collapsed into one element in the new model, such as Name and C_Name into Name. Other than object merge, the security extension of the Merge operator merges access control rules into a new set of access control rules, i.e. ACR_3. The process of merging two access control rules is called *access merge*.

Access merge is based on subject mappings. As shown in Figure 6, Map_a denotes the relationship between all possible subjects of two input access control rules. The two mapped subjects should be collapsed into one subject, such as Customer and Cust into Customer, and share the same access authorization. The access merge is represented by graph transformation as in visual model management [23], and should ensure a safe output by preventing violating accesses while keeping maximum access for subjects.

Suppose the Merge operator takes input models M_1 and M_2. If the mapped subjects have the same access to the same mapped objects in both models, then the related two rules can be merged into one output rule, e.g. Rule 1 of ACR_1 (in Table 1) and Rule 1 of ACR_2 (in Table 2).

Apart from the above case, three other cases need to be handled carefully as follows:

1. Unmapped subjects, subject S_1 for M_1 is not mapped and is a new subject to M_2. Since it is only effective on the elements in M_1, we simply add the related access control rules to the result.

2. Unmapped objects, object O_1 for M_1 is not mapped and is a new object to M_2. We simply add the two rules to the result.

3. Conflict, conflict occurs when a prohibited access is allowed. The following three access conflicts are discussed and corresponding solutions are presented.

a. Allow vs. Deny

Suppose Rule 1 allows subject S_1 the access to object O_1 in model M_1 and Rule 2 denies the access of S_2 to O_2, where S_1 mapped to S_2, O_1 mapped to O_2. Conflict arises when two subjects and their respective objects are merged, i.e. S_1 and S_2 into one subject (called S_3) and O_1 and O_2 into one object (O_3). Whether to allow the access of S_3 to O_3 would be a delicate issue. Possible solutions include:

(1) Deny the subject's access in the resulting rule;

(2) Allow the subject's access in the resulting rule;

(3) Separately create two rules for each subject, and remove the mapping between the two subjects;

(4) Request a user intervention.

Allowing all the access will break the access control rule for M_2. Solution (3) separates mapped objects thus breaks the mapping. Solution (4) requires users' intervention and will produce a result depending on the policy. Users' intervention requires a user interface with the security extension for model management to be user friendly, i.e. a visual environment. Our solution is a hybrid of solutions (1) and (4), i.e. denies the subject's access and requests users' intervention, thus provides safe suggestions that are customizable.

b. Local vs. Recursive

The mapped subjects may have different propagations, e.g. Rule 1 allows access of S_1 to O_1 locally while Rule 2 allows access of S_2 to O_2 recursively. Possible solutions to the conflict include:

(1) Restrict the access to be local in the resulting rule;

(2) Allow the access to be recursive.

Solution (2) gives more access to users than solution (1) does, but can produce possible violation. It would be safe to provide only local access with solution (1).

c. Read vs. Write

A conflict arises if a rule of mapped subjects has different actions to mapped objects, e.g. Rule 1 allows read access of S_1 to O_1 while Rule 2 allows write access of S_2 to O_2. Possible solutions include:

(1) Give only read access in the resulting rule;

(2) Allow write access.

For the similar reason as above, we choose solution (1) as the result.

Overall, our solution assures the maximum safe access for users, and prevents security violation caused by Merge while still being flexible and adjustable by security officers according to the application domains. Table 3 shows the resulting access rules for merged models.

Table 3. Access Control Rules for Merged Model

	Subject	Object	Action	Authorization	Propagation
1	Customer	/Item/	Read	Allow	R
2	Vendor	/Item/	Read	Allow	R
3	Administrator	/Item/	Read	Allow	R
4	Administrator	/Item/Detail	Write	Allow	R
5	Administrator	/Item/Price	Write	Allow	L

5 Discussion and Directions for Future Research

We have discussed access control for model management, and have essentially provided the foundation for work on secure model management.

5.1 Formalization and Other Operators

The access control models discussed here are somewhat informal. The next step is to expand on the work proposed here and develop a formal model and prove that security properties are maintained during the mappings. The access control rules essentially control access that a user can have to the various documents. However a user can receive legitimate responses and subsequently make sensitive associations. Such a problem has come to be known as the inference problem. Extensive work has been carried out on applying security constraint processing for the inference problem [26]. We need to apply intelligent inference to the access rules to achieve more personalized model management.

Other operators also need to extend with security properties, such as ModelGen. After the ModelGen operation, some objects of the original model may be removed, and the security extension of the ModelGen operator needs to adjust the access control rules for the generated model. We will extend other visual model management operators with security properties as our future work.

5.2 Future Work

Future work will proceed in three directions. One is to apply the secure model management for RDF (Resource Description Framework) document. RDF is a critical part of the semantic web. The RDF data model is a syntax-neutral way of rep-resenting RDF expressions. The basic data model consists of three object types, resources, properties, and statements. A RDF model can be represented by a directed graph. Therefore, any node in the RDF model can have multiple children and multiple parents. RDF is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities which enable the automated processing of Web resources. Since RDF is designed to describe the re-sources and the relationship among them without assumption, the definition mechanism should be domain neutral, and can be applied to any domain.

RDF essentially utilizes XML syntax. Therefore, we need to extend the model driven operators for RDF syntax as well as semantics.

The second direction is to extend the concepts for secure information sharing in heterogeneous and federated environments. Organizations are forming coalitions to share data but at the same time maintain security and privacy. We need to integrate the heterogeneous data sources and at the same time enforce the various security policies. The model management approach needs to be examined for secure heterogeneous and federated data integration.

The third direction is to examine other access control policies and models. Notable among them are role based access control (RBAC) and Usage control (UCON) models [21] and [18]. RBAC is about users being allowed aces to object depending on other roles. Usage control model proposed recently subsumes several others models proposed in the literature. UCON consists of six components: subjects and their attributes, objects and their attributes, rights, authorizations, obligations and conditions. Subject must possess rights to access objects. In addition certain obligations have to be met and conditions have to be satisfied. Applying model management for RBAC and UCON needs to be examined.

6 Related Work

Since its first introduction in a vision paper [2], many implementations model management have been presented, such as Cupid [14,15] and SFA [16] as match operator implementations, Merge operator presented by Pottinger *et al* [19]. While most of the approaches only concentrate on individual operators of the model management, Rondo [17] is the first prototype of the generic model management system. None of these proposals addresses security extensions for any model management operators.

Many proposals on access control mechanisms have been presented in both database literature [9,11,10,12,22] and XML area [3,5,7]. There are however few proposals on access controls across heterogeneous data models, and the most related works are those on secure XML federations [27] and XML security models using relational databases [13]. Tan also proposed an idea of using RDBMS to handle access controls for XML documents, in a rather limited setting [25]. Farkas *et al.* developed algorithms to automate the access control rules transformation process, while preserving the Access Control requirements of the original systems [8]. They studied and developed methods for automatically translating Access Control Lists and Bell-LaPadula models to ASL. They concentrated only on the access control rules while our system can manipulate the related schemas at the same time.

In addition, there has been lot of work on access control on temporal models, multimedia models, geospatial information systems and multimedia systems [1]. While these works concentrate on domain-specific access controls, our approach provides security extensions to generic systems and can be applied to virtually any data models.

7 Conclusion

This paper has discussed uniform access control rules for heterogeneous data models and a visual representation of the access control model. We presented approaches for automatic generation of subject matching. We proved that the security isomorphism algorithm generates safe mappings. The paper also discussed the security issues involved in the Merge operator and other operators, and addressed the principles of a secure Merge operator. The security extensions to our previous work on visual model management operators provides automatic generation mechanism for managing access control specifications to allow heterogeneous Web data models to exchange information over public networks.

Model management is becoming an important technology for Web information management. It is critical that security be incorporated into the process at the beginning and not as an afterthought. The major contribution of this paper is attempting to incorporate security into the model management process.

References

1. V. Atluri and S. Chun, An Authiruization, Model for Geospatial Data, *IEEE Transactions on Depoendable and Secure Computing*, Volume 1, #4, 2005.
2. P.A.Bernstein, A. Halevy, and R.A. Pottinger, A Vision for Management of Complex Models, *SIGMOD Record,* 29(4), 55-63, 2000.
3. E. Bertino and E. Ferrari. Secure and Selective Dissemination of XML Documents, *IEEE Trans. Information and System Security (TISSEC),* 5(3): 290 – 331, Aug. 2002.
4. Bray, T., Paoli, J., Sperberg-Mcqueen, C., and Maler, E. Extensible Markup Language (XML) 1.0 (2nd Edition), *World Wide Web Consortium (W3C),* http://www.w3.org/TR/REC-xml, 2000.
5. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, Securing XML Documents. *Proc. EDBT 2000 Konstanz, Germany, Lecture Notes in Computer Science*, Vol. 1777, Springer, New York, March, 2000, 121–135.
6. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, Fine Grained Access Control for SOAP E-Services*, Proc. 10th Int. World Wide Web Conference*, Hong Kong, China, May, 2001.
7. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, A Fine-Grained Access Control System for XML Documents, *ACM Trans. Information and System Security (TISSEC)*, 5(2)169-202, May 2002.
8. C. Farkas, A. Stoica, P. Talekar, APTA: an Automated Policy Translation Architecture, *Int. Conf. Computer, Communication and Control Technologies*, 2003.
9. P. P. Griffiths and B. W. Wade, An Authorization Mechanism for a Relational Database System, *ACM Trans. Database System (TODS),* 1(3): 242 – 255, Sep. 1976.
10. S. Jajodia and R. Sanhu, "Toward a Multilevel Secure Relational Data Model", *ACM SIGMOD*, May 1990.
11. S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino, A Unified Framework for Enforcing Multiple Access Control Policies, *ACM SIGMOD*, 474 – 485, May 1997.

12. S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian, Flexible Support for Multiple Access Control Policies, *ACM Trans. Database Systems (TODS)*, 26 (2): 214 – 260, June, 2001.
13. B. Luo, D. Lee, W. Lee, P. Liu, A Flexible Framework for Architecting XML Access Control Enforcement Mechanisms, Proc. *VLDB Workshop on Secure Data Management in a Connected World (SDM)*, Toronto, Canada, August 2004.
14. J. Madhavan, P. A. Bernstein, and E. Rahm, Generic Schema Matching Using Cupid, *Proc. 27th VLDB Conf.*, Roma, Italy, Sep, 2001, 49-58.
15. J. Madhavan and A. Y. Halevy, Composing Mappings Among Data Sources, *Proc. 29th VLDB Conf.*, Berlin, German, Sep 2003, 572-583.
16. S. Melnik, H. Garcia-Molina and E. Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching, *Proc. 18th ICDE*, San Jose CA, Feb 2002.
17. S. Melnik, E. Rahm, and P. A. Bernstein, Rondo: A Programming Platform for Generic Model Management, *Proc. SIGMOD 2003 Conf.*, San Dieago, CA, June 2003, 193-204.
18. J. Park and R. Sandhu, The UCONABC Usage Control Model, *ACM Transactions on Information and System Security*, Volume 7, Number 1, February 2004.
19. R. A. Pottinger and P. A. Bernstein, Merging Models Based on Given Correspondences, *Proc. 29th VLDB Conf.*, Berlin, Germany, 2003, 826-873.
20. Rahm, Erhard and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching, *VLDB Journal*, 10(4): 334-350, 2001.
21. R. Sandhu, E. Coyne, H. Feinstein and C. Youman, Role-Based Access Control Models, *IEEE Computer*, Volume 29, Number 2, February 1996.
22. R. Sandhu, F. Chen, The Multilevel Relational (MLR) Data Model, *IEEE Trans. Information and System Security (TISSEC)*, 1 (1), 1998.
23. G.L. Song, K. Zhang, and J. Kong, Model Management Through Graph Transformations, *Proc. 2004 IEEE Symp. Visual Languages and Human-Centric Computing*, IEEE CS Press, Rome, Italy, September 2004, 75-82.
24. G.L. Song, K. Zhang, B. Thuraisingham, J. Cao, Towards Access Control of Visual Web Model Management, *Proc. 2005 IEEE International Conf. on e-Technology, e-Commerce and e-Service (EEE-05)*, IEEE CS Press, Hong Kong, March 2005.
25. K.-L. Tan, M. L. Lee, and Y. Wang. Access Control of XML Documents in Relational Database Systems, *Proc. Int. Conf. on Internet Computing (IC)*, Las Vegas, NV, Jun. 2001.
26. B. Thuraisingham. Security Constraint Processing in Multilevel Secure Distributed Systems, *IEEE Transaction on Knowledge and Data Engineering*, Vol. 7, #2, April 1995.
27. L. Wang, D. Wijesekera and S. Jajodia., Towards Secure XML Federations, *Proc. 16th IFIP WG11.3 Working Conference on Database and Application Security*, July 28-31, 2002.