Dynamic Integration Strategy for Mediation Framework

Li Yang (draft)
School of Computer Science

Abstract

The trend in the information area is to provide integrated access to diverse and dynamic information sources. We present a modeling and architectural solution to the problem of data integration from heterogeneous data sources based on the use of wrappers and mediators. The mediator supports integrated query processing by the semantic mapping between the global schema type and the local schema types.

However, mediation poses extensive security problems. Protecting proprietary data from unauthorized access is recognized as one of the most significant barriers to the mediation systems.

Our approach benefits in both flexible strategy of data integration based on semantic mappings and dynamic access control in mediation systems.

Keywords: mediation, heterogeneous, interoperability, XML

1 Introduction

A significant challenge facing the information field in recent years has been the integration of heterogeneous data. Enterprises tend to store represent their data using a variety of data models and schemas, while users want to access data in an integrated and consistent fashion. Moreover, enterprises need to use data sources that do not follow well-structured schemas, nor fit into object-oriented or relational database models. From a syntactic point of view, such data sources contain data that are unstructured or semi-structured, i.e. they don't fit into a regular schema. For example, a source may consist of freeform text; even if the text does have some structure, individual fields may vary in unpredictable ways. For example, persons name can be arranged in large number of fashions. From a semantic point of view, different terminologies used in different data source domains could either indicate the same concept or not. For example, the number in an employee record need not be the same as the number in the context of a street address. In addition, the same information could be organized in different structures in different databases, making the integration task almost impossible. Thus there is a pressing need to provide integrated access to information stored in heterogeneous databases and data sources.

The integration of heterogeneous data sources must also consider security concerns. Once a semantic mapping is established among the client request and sources, it is important to consider whether the client role has the necessary credentials to access the data.

Mediators are typically employed in a situation where the client data model does not coincide with the data model of the potential data sources. A mediator provides a mapping of complex models to enable interoperability between clients and sources. Many mediator systems have been proposed to bridge heterogeneous data sources and requests. A standard mediator language [6] proposal requires support for complex types and semi-structured data, abstract types with methods, the exchange of rules that allow the communication of knowledge between the mediator and source as well as the mediator and the client and the exchange of meta data. A main memory database system presents an in-memory set of data to a client and hides the complexities of sec-
ondary storage access [11]. Constraints allow the specification of facts that have to be considered in the context of many others. Constraint satisfaction is the process of considering all constraints to arrive at a state where all constraints are satisfied. Constraints have been used to support an architecture for user interfaces [5, 10], but also in mediator systems [3] where they provide a more flexible and dynamic type mapping between source and client data models.

In this paper we develop and present a reliable technology that allows flexible query processing over heterogeneous information sources. The key to our approach is the establishment of flexible mappings among the heterogeneous sources that consider the semantic correlation but also seamlessly enforce data security.

2 Related Work

Modern integration systems follow the mediation paradigm presented by Wiederhold [19]. Examples for mediators are TSIMMIS [15], Information Manifold [14], HERMES [1], DISCO [18], Garlic [16] and MMM [3]. The goal of such systems is to permit the exploitation of several independent data sources as if they were a single source, with a single global schema. A user query is formulated in terms of the global schema; to execute the query, the system translates it into subqueries expressed in terms of the local schemas, sends the subqueries to the local data sources, retrieves the results, and combines them into the final result provided to the user. Data integration systems can be classified according to the way the schema of the local data sources are related to the global, unified schema. A first approach is to define the global schema as a view over the local schema; such as approach is called global-as-view (GAV). The opposite approach, known as local-as-view (LAV) consists of defining the local sources as views over the global schema.

In LAV, a local change to a data source can be handled locally by adding, removing or updating only the view definitions concerning this source, but the query on the global schema needs to be reformulated in the terms of the local data sources’ schemata; this process is traditionally known as “rewriting queries using views” and is a known hard problem [1]. We use global as view, becaue the use of a global view coincides with the desire of many to integrate information from a variety of sources and present it in an integrated view for a user. Since we map each source to the target independently, changes in a particular source affect only the mapping of that source to the target. Thus, the approach is scalable and can be applied to an environment such as the Web where scalability is paramount. In the integration of heterogeneous databases, our work features in integrating the data sources based on semantic mapping.

One paragraph for mediation security solutions here. Dawson [8] associated wrappers a mapping between the source’s security lattice and other lattices. did not support object granularity.

The paper is organized as follows: Secsion 2 discusses the related works. Section 3 introduces a flexible mediation framework designed to integrate information from heterogeneous data sources. Section 4 illustrated the dynamic integraton strategy in the mediation framework with the security enforced in the data integration. Section 5 presents our concluding remarks

3 A Mediation Framework

With the purpose of integrating data with rich semantics, Ege et. al.[12] propose a mediator architecture for the information integration from heterogeneous databases with the knowledge of the capability of participating sources.

3.1 Architecture of Mediator

The proposed mediator architecture is to handle requests from a user (as in Figure 3). These mediators will play intermediate roles between users and data sources, and these mediators help the users to establish streams to and from the heterogeneous data sources.

The framework features three layers: presence, integration and homogenization/connector. The upper level is the presence layer that makes the data source seem ever-present to the user and communicates directly with the user. The presence layer is responsible to translate the heterogeneous request from user to XML format, extract the data type of request represented by XML schema, and translate the response from the XML format into the original user request format. The presence layer makes it feasible that the
mediation architecture handles the request from any kind of devices or in any kind of formats by the two-way format translation. Therefore the work of underlying layers is encapsulated.

The middle integration layer resolves the schema differences between the user needs and the source availability by schema mapping [2]. The entities in the integration layer are the “Mediator_composers” who are able to decompose the schema if necessary and locate the destination data source for a specific schema. Upon every request from a client, one global mediator will be elected from the “Mediator_composers” based on the availability. Before the election, the relationship between the mediator_composers are peer-to-peer. After the election, the global mediator distribute the query to the relevant mediator_composers and composed the retrieved information from the mediator_composers. This process of global mediator election dynamically determines the hierarchical structure in the integration layer for each request. This procedure makes the architecture more adaptive to both the network capability and mediator load, and then more efficient for the multimedia data operation, i.e. streaming, than a fixed architecture. The bottom level homogenization layer makes heterogeneous data sources appear to have a unifying XML schema.

With the assumption that the query to the databases happen more frequently than the update of the databases, the mediation systems include two phases as follows. This paper we put the main efforts on the second run-time query answering phase.

**Initialization/Preparation phase**

1. Each source has its own schema type and advertise to the global level. A global schema type was generated by the methods proposed in current related works [4].

2. Each source maps to the global schema type. By the mapping, each sources generates it capability against global schema type. That is each source pushes its contribution to global mediator.

3. Global mediator maintains the sources capability lookup for all sources.

**Run-time Query Answering phase**

The following steps refer to the message passing in the architecture of mediation showed in Figure 1.

1. User log in to the system and messages are sent to the Session Initiation Server.

2. Session Initiation Server (SIS) elects the global mediator and returns the secure visible global view (SVGV) to the current user. The details in SIS will be discussed in section 4.2.

3. User poses query against the specific secure global schema type.

4. Elected global mediator distribute the query to the relevant mediator_composers that can answer the queries.

5. Relevant mediator_composers translated queries represented by the global schema type, into queries represented by the local schema types.

6-7 The relevant sources execute the queries, and return the results to the corresponding mediator_composers.

8-9 The mediator_composers translate the results represented by the local schema type into the ones represented by the global schema type.

10 Global mediator integrates the results and returns to the user.
3.2 A Motivating Example

This section we introduce a motivating example that helps to illustrate the dynamic integration among the heterogeneous databases. Assume in a hospital system, patients’ medical records are stored in the different departments whose databases may be heterogeneous. The hospital organization maintain a global schema type semantically generated from the schema type of the local sources, in order to support the interoperability and data sharing inside the hospital. The nodes marked same color from the global schema type (Figure 2a) and local schema type (Figure 2b, 2c) denote that the nodes have semantic correspondence. Each mediator_composer maintains a virtual global schema type structured as a tree whose nodes record the relevant source ID that stores the relevant information, indicated as $s_1$ and $s_2$ below the nodes in the global schema type.

Note: Where to store the join information, that is how to integration the sources from two databases based on the common key? Such information is stored in the global mediator. Besides the storage of the source destination for each node, the global mediator also records the relationship between data sources.

The nurse view and doctor view in the Figure 2(a) contains the set of the nodes that different kind of users (nurse, doctor) are allowed to access, which will be illustrated in Section 4.1.

Figure 2: A Motivating Example

4 Dynamic Integration Strategy

This section explains how to initiate a query session from the SIS, how to specify the semantic mappings between the global schema type and local schema type, and how to evaluate the query based on the semantic mappings.

4.1 Session Initiation

The session initiation server module receives the login information from the specific user and returns a visible global view against which the user pose query. Four components are included: the authentication component is used to authenticate the user and capture the user’s profile; the election component is used to select a global mediator with the least load from the mediator_composers; the view computing component is to create a secure visible global view (SVGV) for the current user based on the authentication information, and the authorization policy repositories store the access authorizations that states a subject can (or cannot) access an element or (set of them).

Figure 3: Session Initiation

We introduce the term secure visible global view that denotes part of the global schema type that is visible or authorized to the current subject based on his/her profile and authorization policies. A user could login in to the authentication module with his/her id or on behalf certains roles [17]. We can them subject in general. The idea of view computing [7] is to create and maintain a separate view for each subject (user, role) who is authorized to access a specific portion of global schema type (XML structured in nature). The view contains exactly the set of data nodes that the user is authorized to access. After the view is constructed, during the run time, users can simply run their queries against the views.
without worrying about security enforcements. The view of a subject on the global schema type depends on the access permissions and denials specified by the authorization and their priorities. Such a view can be computed through a tree labeling process [7]. To be more efficient, the visible view will not be calculated upon each request, the view for a specific user or role (nurse, doctor) can be cached in the session initiation server for the later requests, thus improving the efficiency while achieving the flexibility. By the above methods, the data security for heterogeneous sources are seamlessly enforced in our mediation framework.

Because we have multiple users and applications to retrieve and integration information simultaneously, we try to distribute the load among the mediator_composers. In our previous work[12], based on the sensors in the systems, the mediator_composer with the least load is elected as the global mediator that has the knowledge of the global schema type. So after the user authentication, the election component elects a global mediator and exports the global schema type to the view computing component. The view computing component interact with the security policy repositories and outputs the secure visible global view.

4.2 Semantic Mapping Specification

In order to achieve unified identification of mediation object, the transformation from heterogeneous data model to homogeneous data model is performed by wrappers. But the semantic heterogeneity still exists in both terminology and structural aspects. A mediation system is defined and how to resolve the semantic gap among heterogeneous sources is elucidated. In order to support integrated query processing, our mediation system includes three parts: a global (mediated) schema type, a set of source schemas type and mapping relation. A global schema type is tree type whose labels are terms of a global vocabulary, distinct from source schema type used in the labels defining local data schema. The global schema vocabulary has been chosen to unify the local vocabulary and represent a specific domain of interest.

Definition 4.1 (Mapping Relation) between global and local schema type Let $G$ be a global schema type related to a set of $L$ of local schema type. A mapping relation $M$ between $G$ and $L$ is a relationship between $G$ and $L$: $M \subset G \times L$. $g_i$ and $l_i$ are the paths in the global schema type and local schema type respectively, we will denote a pair $(g_i, l_i)$ in $M$ by: $g_i : f_i(\Sigma_i)$, where $f_i$ is the functions used to load and integrate data from the local level to the global level.

This definition is inherited from [9] that defines the mapping between the path in global schema and the path in source schema. We enhanced their definition by the mapping function that includes complex operations [20], like merge, concatenate, thus improving the capacity for semantic gap resolution. We use tree to represent both global and source schema in $M$. A tree includes set of leaf object set $O$ and a set of path relationship $R$. A source-to-global mapping $M_i$ for source schema $S_i$ with respect to a global schema $G$ is a function $f_i(\Sigma_i) \rightarrow \Sigma_G$. Intuitively, a source-to-global mapping $M_i$ represents inter-schema correspondences between a source schema $S_i$ and a global schema $G$. A source-to-global mapping between the two schemas includes a semantic correspondence.

For instance, mediator 1 is used to specify the mapping relationship between the global schema and the source 1 information database (source). It declares that the concatenation of values in $\text{firstname}$ and $\text{lastname}$ in source 1 semantically corresponds to the values of $\text{name}$ in the global. And $\text{patient}/\text{case}/\text{xray}$ in source 1 corresponds to $\text{patient}/\text{record}/\text{test}$ in the global.

patient/personal/id @ g := patient/id@s1
patient/personal/name @ g := cont(patient/firstname, 
patient/lastname)@s1
patient/record/test @ g := patient/case/xray@s1

And mediator 2 is used to specify the mapping relationship between the global schema and the source 2 information database (source). It specifies that the node name of female or male corresponds to the values of the $\text{gender}$ in the global. The $\text{prescription}/\text{medicine}$ and $\text{case}/\text{disease}$ in source 2 map to the $\text{prescription}/\text{treatment}$ and $\text{record}/\text{diagnosis}$ in global respectively.

patient/personal/id @ g := patient/id@s2
patient/personal/name @ g := patient/name@s2
patient/personal/relative @ g := patient/relative@s2
patient/personal/gender @ g := label(patient/gender/female, 
patient/gender/male)
patient/prescription/treatment @ g := patient/prescription/mecidine@s2
patient/record/diagnosis @ g := patient/case/disease@s2
4.3 Query Evaluation

The users queries are issued with the vocabulary defined in the secure visible global view, and the query is received by the elected global mediator. The global mediator distributes the queries to the relevant mediator_composers that connect to the relevant data sources. When a user log in and authenticated as a doctor, his/her secure visible global view is the schema type enclosed in the outer rectangle in Figure 2(a), called doctor view. The source $s_1$ and $s_2$ become the relevant sources and the mediator 1 and mediator 2 specified in section 4.2 becomes the relevant mediator_composers because those two mediators provide the mapping specification from the doctor view to the local schema type.

For instance, the doctor poses the query $Q$ to retrieve the patients record with the knowledge of patients name. We call the paths that either used to input condition, i.e. patient/personal/name, or the paths used to output query result, i.e. patient/record as the essential paths in the global query tree. To be executable, queries must be translated into queries expressed in terms of vocabulary in the local schema type. Query translation relies on the semantic mapping relations between the global schema type and the local schema types. In the above example, after the global mediator receive the query $Q$, it detects that mediator 1 and mediator 2 are relevant mediator_composers. These two relevant mediator_composers will translate $Q$ into the queries expressed in the local vocabulary based on the semantic mappings.

**Definition 4.2 (Query Translation)** Let

$$Q = \{p_g\}$$ where $\{p_g\}$ be a set of path from the global tree query $Q$. The query translation $Trans$ returns the set of path $\{p_l\}$ in the vocabulary of local schema type with the input of the mapping relation $M$ and the global tree query $Q$, which is represented as $\{p_l\} = Trans(\{p_g\}, M)$.

In the query translation process, the paths in the global tree query are search in the mediator specifications and translated to the local tree queries. With the knowledge stored in global mediator that the information from source 1 and source 2 can join on patients id. The id becomes the essential path also. So the essential paths in $Q$ are: patient/personal/id, patient/personal/name, patient/record/test and patient/record/diagnosis, and the corresponding in mediator 1 is patient/id, patient/firstname, patient/lastname and patient/case/xray. Correspondingly, the query $Q$s essential paths set in source 2 includes: patient/id, patient/name and patient/case/disease. The complexity of the translation depends on the number of the essential paths in the global query tree and the number of semantic mappings in the mediator specifications. Finally, the answers from the source 1 and source 2 are joined to serve the user. Note, the patient id here can assure that the information from the same person in the real world is integrated.

5 Conclusion & Future Work

In this paper we presented how to integrate information from heterogeneous sources while supporting flexible security enforcement in mediation systems.

In our future work, the probabilities are associated to the semantic mappings in the mediator specification in order to support flexible query answering. The ranked answers are returned to the users based on the probabilities from the mappings.

References


