A Comprehensive Framework Towards Information Sharing Between Government Agencies

Akhilesh Bajaj, The University of Tulsa, USA
Sudha Ram, The University of Arizona, USA

ABSTRACT

Recently, there has been increased interest in sharing digitized information between government agencies, with the goals of improving security, reducing costs, and offering better quality service to users of government services. The bulk of previous work in interagency information sharing has focused largely on the sharing of structured information among heterogeneous data sources, whereas government agencies need to share data with varying degrees of structure ranging from free text documents to relational data. In this work, we explore the different technologies available to share information. Specifically, our framework discusses the optional data storage mechanisms required to support a Service Oriented Architecture (SOA). We compare XML document, free text search engine, and relational database technologies and analyze the pros and cons of each approach. We explore these options along the dimensions of information definition, information storage, the access to this information, and finally the maintenance of shared information.

Keywords: databases; digital government; e-government; heterogeneous databases; information sharing; semantic conflicts; semantic resolution; XML

INTRODUCTION

The digitization of information has fundamentally altered the environment in which government agencies conduct their missions and deliver services. Recently, there has been considerable interest in exploring how emerging technologies can be used to promote information sharing among different governmental agencies (Bajaj & Ram, 2003). Such information sharing is desirable for several reasons. First, increased levels of security can be achieved if different government agencies share information. These effects can be felt in areas as diverse as global counter-terrorism (Goodman, 2001) and the war on drugs (Forsythe, 1990). Several recent articles, for example, (Dizard, 2002), strongly endorse the view that the sharing of intelligence information amongst dif-
ferent law enforcement agencies will enhance their ability to fulfill their required functions. Second, there has been a growing need to streamline inter-agency communication from a financial savings perspective. For example, Minahan (1995) shows how the lack of information sharing between different government organizations considerably hampered the establishment of an import-export database that would have streamlined the flow of goods into and out of the US and potentially saved billions of dollars. As pointed out in (Stampiglia, 1997), data sharing between health care agencies can also result in significant cost savings. Third, inter-agency information sharing results in offering fewer contact points for end-users of public services, thereby leading to more efficiencies in the delivery of these services to the end-users. For example, allowing agencies to share geographic information systems (GIS) information improves the quality of customer service afforded to end-users of these services (Hinton, 2001). Other common examples of activities that can benefit from information sharing include: the application for licenses and permits and the ability of aid workers to provide essential services.

Much work has been done in the area of the integration of structured information between heterogeneous databases (Hayne & Ram, 1990; Reddy, Prasad, Reddy, & Gupta, 1994; Larson, Navathe, & Elmasri, 1989; Batini, M. Lenzerini, & Navathe, 1986; Hearst, 1998; Ram & Park, 2004; Ram & Zhao, 2001). The two broad approaches in this area are a) the creation of virtual federated schemas for query integration (Zhao, 1997; Chiang, Lim, & Storey, 2000; Yan, Ng, & Lim, 2002) and b) the creation of actual materialized integrated warehouses for integration of both queries and updates (Vaduva & Dittrich, 2001; Hearst, 1998). While the area of structured information integration is relatively well researched, considerably less attention has been paid to the area of the integration of unstructured information (e.g., free text documents) between heterogeneous information sources. Recently, several researchers (Khare & Rifkin, 1997; Sneed, 2002; Glavnic, 2002) have pointed out the advantages of the XML (extensible markup language) standard as a means of adding varying degrees of structure to information, and as a standard for exchanging information over the Internet.

As pointed out in (Dizard, 2002; Minahan, 1995; Stampiglia, 1997), much of the information that government organizations share is at least somewhat unstructured. The primary contribution of this work is a comprehensive framework in the service oriented architecture (SOA) context, that explores three diverse technologies that exist to enable information sharing: digitized free text documents with related search engines, relational database management systems, and XML document repositories. We explore how SOA allows for a) the ability to provide varying degrees of structure to the information that needs to be shared, by sharing all information in the form of XML documents and b) the inclusion of various groups’ viewpoints when determining what information should be shared and how long it should be shared. We also compare methods of storing and accessing the information under the different technologies. We compare the three technologies using well-understood criteria such as ease of information definition and storage, ease of information access, and ease of system maintenance.

The rest of this paper is organized as follows. In the second section, we discuss prior research in the area of information integration. In the third section, we describe SOA and data integration, and describe the potential conflicts that would need to be resolved in order to arrive at common data definitions. In the fourth section, we present alternative strategies for information definition as well as mechanisms for information storage and retrieval and compare these alternate methodologies. The fifth section contains the conclusion and future research directions.

**PREVIOUS WORK**

Much of the work in the area of information integration has focused primarily on integrating structured data from heterogeneous sources. Excellent surveys of data integration strategies are presented in (Batini et al. 1986;
Hearst, 1998; Chiang et al. 2000). There have been primarily two broad strategies used to integrate structured data: a) retain the materialized data in the original stores, but use a unified federated schema to allow the querying of heterogeneous sources or b) actually materialize the combined data into a unified repository to allow for faster query response and also allow updates. Note that b) requires a unified federated schema also, but the schema in this case is not virtual, as in a).

Work in the area of unified federated schema generation is well established (e.g., Batini et al. 1986, has an excellent survey of early work in the area). Several issues have been addressed in this area. First, Blaha & Premelani (1995) highlight the commonly observed errors in the design of the underlying heterogeneous relational databases, which often need to be resolved before integration is possible. Second, the issue of semantic inconsistencies between database object names (such as attribute names) has been widely addressed. Strategies to resolve semantic conflicts have ranged from utilizing expert systems (Hayne & Ram, 1990) to neural networks (Li & Clifton, 1994). Recently, several researchers have recognized this problem to be only partially automatable (Chiang et al. 2000). A recent solution to partially automate semantic resolution (Yan et al. 2002) utilizes synonym sets, with similarity measures. A set of potential unified federated schemas is generated using algorithms proposed in this work, and final selection of the unified schema is done manually. As an alternate solution, a schema coordination methodology is proposed (Zhao, 1997), where the minimal mapping is done only at the semantic level (rather than at the logical level) and overheads are lower than in traditional schema integration. However the tradeoff is that this methodology can only be used for querying, and the query resolution process is more complex than with a federated schema.

In the area of materialized data integration, a federated schema is required as a first step. However, there are several additional issues such as the retention of legacy systems, the coordination and refresh rate of information in the materialized warehouse, and the resolution of data quality issues (Chiang et al. 2000; Vaduva & Dittrich, 2001). Chiang et al. (2000) highlight possible problems that can arise when integrating actual data, after the schema integration has taken place. Examples of these problems include entity identification, relationship conflicts, and attribute value discrepancies between data from heterogeneous sources. Even though much work has been done in the area of the integration of structured information, Silberschatz, Stonebraker, & Ullman (1996) highlight it as one of the major research directions for database research in the future, and work continues in this area.

Unlike the integration of structured information, considerably less work has been done in the area of unstructured information integration. The domain of interest in our work is the sharing of information between government organizations. This raises several new issues. First, while traditional integration work has considered domains where there are heterogeneous structured database schemas, much of the information shared between government organizations tends to be unstructured (Dizard, 2002; Minahan, 1995; Stampiglia, 1997). As such, structured data integration methodologies are insufficient to allow data sharing between government organizations. Second, since information in government organizations is often sensitive from a policy and privacy perspective, the actual definition of the information that needs to be shared is often performed by several parties. Thus, applicable methodologies in our domain of interest should allow various groups to flexibly structure information at varying degrees of structural rigidity. Thus, for example, certain information (such as names and addresses) can be structured down to the same detail as relational database columns are structured, while other information (such as descriptive comments, rules, and regulations) should be retained as free text.

Recently, several researchers have pointed out the advantages of the XML standard as a means of adding varying degrees of structure to information and as a standard for exchanging information in different domains. For example,
Sneed (2002) points out how XML can be used to pass data between different software programs in batch or real-time mode. Glavinic (2002) describes how XML can be used to integrate applications within an organization. Next, we explore the different options for data integration within an SOA context, in order to allow the sharing of information between heterogeneous information sources. This is useful since information sources in a government organization can range in structure from repositories of text documents to relational databases.

**DATA INTEGRATION OPTIONS**

**Core Design Criteria Behind SOA**

The goal of SOA is to achieve loose coupling among interacting software programs. A service is a unit of work performed by a service-provider software for a consumer software (He, 2003; Pazoglou and Georgakopoulos, 2003). Services are invoked by consumers or client applications via messages that conform to descriptive schemas. Service descriptions are usually exported by providers to registries. Consumers contact a broker software that helps discover the service. SOA is different from standard client/server architecture in that a schema defines the functionality for each service and registries are available for lookup and binding without knowledge before the execution (Vetere and Lenzerini, 2005). One example of a service provider may be a relational DBMS that provides read and write access to its fields, with consumers using these services to actually make changes to the data. In SOA, the semantic layer is used to ensure that the data embedded within messages are interpreted in a similar fashion by consumers and producers. For example, attribute `customer_id` may be a unique identifier in one database, whereas `customer_name` and `customer_address` together may be the identifiers in another database. If a client supplies the `customer_id`, `name` and `address`, and asks for all the customer service conversations related to a certain customer, from two different databases, different web services may be needed to be run on the two databases.

As pointed out in (Vetere and Lenzerini, 2005), the semantic layer includes operational aspects that characterize the provider’s service. It covers events and states, in effect the overall ontology of the application. Figure 1 shows a web services framework for data visibility, that utilizes an ontology layer for semantic resolution. A service request can be translated into appropriate requests based on each target database, based on the ontology layer.

**Components of a Data Services Infrastructure**

We consider three major components important to defining data services: a) an information definition component, b) an information storage component, and c) an information retrieval component.

---

**Figure 1. Web Service Layers for Data Services, adapted from Ceccola (2006)**

![Diagram of web service layers for data services](image-url).

---

Copyright © 2007, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
Information Definition

Several choices are available for specifying how the information to be shared among government agencies can be defined. We list three commonly used standards here. The first and most obvious is free text documents, where the name of the document along with other fields such as keywords can be used to structure information about the document. This standard includes documents in pure text markup languages such as HTML (hypertext markup language). Free text documents are appropriate when storing records of proceedings, hearings, rules and regulations, or memoranda. A significant portion of government information is in this format (Dizard, 2002; Minahan, 1995; Stampligia, 1997).

The second standard is using a relational database structure (Codd, 1970), and it involves storing information in the form of relations, where each table has attributes and the tuples contain the actual data. Relations are linked to each other using common attributes that usually uniquely identify a tuple in one table. The relational model has well-accepted rules for eliminating data redundancy using the concept of data normalization. Most commercial database management systems (DBMSs) support this standard. This option is useful if the information to be shared consists of clearly defined attributes such as: the name, address, physical description and attributes describing immigration entries and exits for individuals.

The third standard for structuring information is the use of XML document type definitions (DTDs)\(^2\). XML DTDs allow information to be structured, using tags to differentiate elements and sub-elements. This is analogous to tables and columns in the relational model. Several researchers believe that XML will allow the Web to become more semantic in nature, such that software agents will be able to access information and understand what it means (Berners-Lee, Hendler, & Lassila, 2001). The main advantage of XML are a) it is a platform independent standard where the data does not need to be read by a proprietary product (as it would in a relational DBMS) and b) it allows for variable structuring of information, where some parts can be free text and others can be more structured.

When creating an XML DTD, the first step is for the players involved to create fields in each DTD, and also set limits on the amount of time information items are shared. For example, if a county’s police and treasury departments wish to share information, they will first collaborate and create DTDs of the information items they wish to share. These DTDs will be discussed and agreed upon by the various players described earlier, until a settlement is reached. An example of two such DTDs is shown in figure 2. The DTD in figure 2a shows information that the police agrees to provide to the treasury department if the latter desires to verify if a specific business license applicant has been convicted of a felony. The information on each felon contains the name (first and last), the social security number, a list of convictions, and an expiry date. The convictions list has repeatable conviction items. PCDATA stands for parsed character data (text) while the * symbol indicates that the item can be repeated. The DTD in figure 2b is information that the treasury department agrees to share with the police to notify them of business applications that are being processed. The expiration date tag allows a software program to delete elements whose expiration date has passed.

Data and Schema Conflict Resolution Strategies

A major impediment to creating DTDs that span agencies, as well as populating the actual XML pages that contain data gathered from databases in different agencies, is the resolution of conflicts that exist between the data and data-definitions that resides in the different databases. In this section, we identify several types of conflicts that can occur.

Based on work presented in (Ram, Park et al. 1999), we divide conflicts into two main types: data-level and schema-level. Data-level conflicts are due to differences in data domain and storage decisions, while schema-level con-
Conflicts occur because of differences at the logical definition (schema) level. Next, we describe the different sub-types of these conflicts.

a. Data-Level Conflicts:
1. **Value Conflicts**: These occur because the same data values (for the same attributes) can have different meanings that are context dependent. For example, the value "excellent" in one context for a convict's rating may mean the convict did nothing negative, while in another database, the value "excellent" may mean they were proactive in good behavior.

2. **Representation Conflicts**: These arise when similar real-world objects are described using different data types or format representations. For example, the *expiration date* for a conviction record may have different formats in different databases.

3. **Unit Conflicts**: These are due to different units being used to represent the same data. For example, if data on the length of parole of a felon was being shared, it could be weeks in one database, and months in another.

4. **Precision Conflicts**: These occur when data is represented using different scales or granularities. For example, the population of prisoners may be stored in tens in one database, and in hundreds in another. Similarly, the satisfaction rating of parole boards with a prisoner may be captured using a 3 point scale in one database and a 5 point scale in another, leading to different granularities of information.

b. Schema-Level Conflicts
1. **Naming Conflicts**: These are due to assignment of different labels for schema objects such as attributes, entity sets, and relationships sets. Homonyms occur when the same word is used for two different meanings in different databases. For example, "Inspector" in the police database versus "inspector" in the treasury database. Synonyms are two or more words, used to describe the same concept. For example, "prisoner" in one database and "innate" in another database. Homonyms and synonyms can also occur at the entity and relationship levels.

2. **Primary Key Conflicts**: These arise when the same set of objects is identified using different sets of properties across databases. For example, a prison may be identified by its social security number in one database, and his *prison_id* in another database.

3. **Generalization Conflicts**: These occur when different generalization choices are made across databases. For example, one database may have an entity set called "prisoners" that includes all prisoners. Another database may have three different entity sub-classes called "maximum security prisoners", "medium security prisoners", and "minimum security prisoners".

4. **Schematic Discrepancies**: These occur when a different schema is used to represent the same information. For example, a citizen's application for a business license can be either an entity by itself, or a relationship between the *applicants* entity set and the *business_license_types* entity set. In other words, the same information can be structured differently across databases, based on the schema designs.

The different types of conflicts described need to be resolved as part of the information definition phase. Two options exist to accomplish this. First, recent work on automated conflict resolution such as in (Ram and Park 2004) can be used to automate a large portion of conflict resolution. Second, a one-time manual walk-through can be used to understand the conflicts between different databases, after which translation programs can be written so that the databases deliver information in the form of XML pages conforming to the DTDs that are finalized in the information definition phase.

**Information Storage**

Based on the three alternate standards of information definition described, there are alternate methods of storing the structured infor-
Figure 2. Example of DTDs created to share information between Police and Treasurer.

```xml
<?XML version = "1.0" ?>
<DOCTYPE DOCUMENT [
  <ELEMENT DOCUMENT (FELON)*>
  <ELEMENT FELON (NAME, SSN, CONVICTIONS, EXPIRATIONDATE)>
  <ELEMENT NAME (LASTNAME, FIRSTNAME)>
  <ELEMENT LASTNAME (#PCDATA)>
  <ELEMENT FIRSTNAME (#PCDATA)>
  <ELEMENT SSN (#PCDATA)>
  <ELEMENT CONVICTIONS (CONVICTIONITEM)*>
  <ELEMENT CONVICTIONITEM (CONVICTIONDATE, CONVICTIONDESCRIPTION, CONVICTIONSENTENCE)>
  <ELEMENT CONVICTIONDATE (#PCDATA)>
  <ELEMENT CONVICTIONDESCRIPTION (#PCDATA)>
  <ELEMENT CONVICTIONSENTENCE (#PCDATA)>
  <ELEMENT EXPIRATIONDATE (#PCDATA)>
]>

a) DTD of Information Shared from Police to Treasurer.

```xml
<?XML version = "1.0" ?>
<DOCTYPE DOCUMENT [
  <ELEMENT DOCUMENT (APPLICANT)*>
  <ELEMENT FELON (NAME, SSN, APPLICATIONS, EXPIRATIONDATE)>
  <ELEMENT NAME (LASTNAME, FIRSTNAME)>
  <ELEMENT LASTNAME (#PCDATA)>
  <ELEMENT FIRSTNAME (#PCDATA)>
  <ELEMENT SSN (#PCDATA)>
  <ELEMENT APPLICATIONS (APPLICATIONITEM)*>
  <ELEMENT APPLICATIONITEM (APPLICATIONDATE, APPLICATIONDESCRIPTION, APPLICATIONDECISION)>
  <ELEMENT APPLICATIONDATE (#PCDATA)>
  <ELEMENT APPLICATIONDESCRIPTION (#PCDATA)>
  <ELEMENT APPLICATIONSENTENCE (#PCDATA)>
  <ELEMENT EXPIRATIONDATE (#PCDATA)>
]>

b) DTD of Information Shared from Treasurer to Police

```

Information as well. If the information is structured in the form of free text or HTML files, then storage on a file server (usually a component of the operating system) is a first alternative. Another option is third party document management tools such as Documentum (www.documentum.com), that facilitate the creation and management of these pages. Most of these systems use the existing file system to store the applications, but store additional information that makes it easier to retrieve the files. Finally, it is possible to use object-relational features of current commercial DBMSs to store and retrieve these files.

For information stored in the form of free text, search engines, for example, Google and AltaVista, offer one method of accessing information. This is similar to a search on the Internet, except that instead of searching across numerous web sites, the search in our case is on a repository of free text or HTML documents. A second method is to use directory based search engines such as Yahoo (www.yahoo.com). If a third party application is used for storage, the
search capabilities are limited to the capabilities of the tools offered by that application. Finally, if object-relational DBMSs are used, the ease of access depends on extensions to the structured query language supported by the DBMS.

If the information is structured in a relational database, a commercial DBMS can be used to store this information. Commercial relational DBMSs are a tested technology that allow for concurrent access by multiple users. They provide good throughput, recovery and reliability for most applications. The structured query language (SQL) standard allows easy access to information. Results can be served on the Internet using a number of available technologies such as active server pages (ASP) or JSP.

If the information is structured in XML, it consists of text files, though the text is now structured with tags. Storage alternatives for XML files are similar to free text, that is, file systems, third party applications, and object relational DBMSs. However, there is an important difference between free text and XML formats, when accessing the information. For XML files, the methodology of using conventional search engines with key word searches either breaks down, or at the very least, negates the whole purpose of structuring the information as an XML document. Similarly, directory based search engines do not allow the search and retrieval of portions of XML pages. To overcome these problems, two standards have been proposed by the WWW consortium (www.w3c.org) to search XML files. These are the XPATH standard, and the more recent XQUERY standard, which is still evolving. As of now, these standards allow the querying of a single XML page, not a repository of XML pages.

In order to create an XML repository, first, a one-time effort is required to create reports from the local databases or document repositories (of the police and the treasury) in the form of XML pages. Since most commercial DBMSs currently offer the ability to create scheduled reports in XML, this task is possible without adding to existing IS systems and personnel. Next, a repository is created to store these pages. This repository is a directory structure with an IP (internet protocol) address that consists of interlinked HTML and/or XML pages. Finally, procedures are implemented to affect the periodic transfer of pages to this repository, and purging of these pages from the repository.

Figure 3 summarizes the different methods of information storage (in each column), and the alternate methods available to access them (along the rows).

**Information Retrieval**

Information retrieval of free text documents can be performed with text search

---

**Figure 3. Information formats and their possible methods of search**

<table>
<thead>
<tr>
<th>Information Format →</th>
<th>Free text /HTML Pages</th>
<th>Relational Alphanumeric Data</th>
<th>XML Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Mechanisms ↓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search Engines using Key Words</td>
<td>X</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Directory Based Search Engines</td>
<td>X</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Third party applications for storing and searching pages</td>
<td>X</td>
<td>NA</td>
<td>X</td>
</tr>
<tr>
<td>Structured Query Language</td>
<td>NA</td>
<td>X</td>
<td>NA</td>
</tr>
<tr>
<td>XPATH, Xquery for single XML pages</td>
<td>NA</td>
<td>NA</td>
<td>X</td>
</tr>
</tbody>
</table>

Copyright © 2007, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
engines such as www.google.com or www.yahoo.com. Information stored in relational databases can be retrieved using SQL. For XML document repositories, one example of a search agent is XSAR (XML search agent for information retrieval) that searches for information from the repository (Bajaj, Tanabe, & Wang, 2002; Bajaj and Ram, 2003). XSAR can dynamically query large information repositories of XML documents. It is **DTD independent**, so the same agent can be used to query documents consisting of multiple DTDs. XSAR is also a dynamic agent, in that it does not use an underlying database or directory scheme of information on pages. Rather, it dynamically queries the repository on behalf of a user. There is no requirement for the repository site to have any special software or hardware; the only assumption is that it is a site that contains a mixture of HTML and XML pages.

XSAR does require the user to be aware of the underlying DTD, insofar as being aware of which fields the user needs to search. Essentially, XSAR reformulates the query as an XQL expression, and then launches a spider that traverses the information repository and executes the XQL query against each XML page found in the repository. Figure 4 shows the operational flow of XSAR.

**Choice of XML Query Language**

Currently, there is no standard query language for XML designated by the WWW Consortium (W3C). However, several query languages for XML documents have already been proposed, for example, **XQL**, **XML-QL**, and **Quilt** (all accessible via http://www.w3c.org). XQL was utilized when implementing XSAR because a) its grammar is based on Xpath, which has already been standardized by W3C and b) application programming interfaces (API) for XQL are available in Java.

**COMPARING THE POTENTIAL METHODOLOGIES**

**Ease of Information Definition and Storage**

In the **free text** methodology, unstructured information such as agency documents can be identified and shared. However, it is not possible to define specific, structured information that can be shared. Since most agencies have a collection of documents, the information definition step will in general be the easiest in this phase, and consist of a relatively simple identification of documents that can be shared, without any in-depth study of the actual data el-

---

*Figure 4. Operational flow of XSAR*
ments in the agencies. In the database schema methodology, the schema of each government agency will have to be prepared for review by the different parties. Different data items can then be identified for sharing. A similar process can be followed for XML document repositories, where the data elements will need to be defined and then a DTD created. Thus, both the database schema methodology and the XML repository methodology require an examination of the actual data elements used by the agencies.

From the storage standpoint, the free text methodology requires that the pages be stored on a server's file system. A coordination mechanism is needed in place so that changes to documents in the organizations' internal repositories are reflected in the shared repository. In the database schema methodology there is no need for additional storage of data, since information is obtained dynamically from the underlying databases within each organization. With XML repositories, storage is similar to the free text methodology, with a coordination mechanism to ensure that the pages are refreshed periodically to reflect the structured information within the internal databases. Thus, the database schema methodology requires less implementation effort from a storage perspective, than with free text and XML repositories.

From this discussion, in general, we see that the free text methodology offers the easiest data definition while the database schema methodology offers the easiest information storage.

Ease Of Information Access

Web search engines that search files on different web sites have their origin in the information retrieval (IR) systems developed during the last 50 years. IR methods include Boolean search methods, vector space methods, probabilistic methods, and clustering methods (Belkin & Croft, 1987). All these methods are aimed at finding documents relevant for a given query. For evaluating such systems, recall (ratio of the number of relevant retrieved documents to the total number of available relevant documents) and precision (ratio of the number of relevant retrieved documents to the number of total retrieved documents) are the most commonly used measures (Billus & Pazzani, 1998). Finally, response time (time taken for the user to get the information she desires) has been found to be a useful metric (Anderson, Bajaj, & Gorr, 2002).

Furthermore, the performance of each search tool is directly influenced by the user's ability to narrowly define the nature of the query, for example, in the form of precise key-word strings or correct specification of a search condition. For the sake of the discussion below, we assume that the search specificity is held constant.

Recall

For the free text methodology, recall is the ratio of the number of pages identified correctly as matching to the total number that should have been identified as matching. This ratio is clearly dependent on the algorithms used to perform the matching, with most current algorithms based on weighted indices. In general the ratio will be less than one. Also, as the number of pages increases, all else being equal, the recall drops.

For the database schema methodology, the recall of a search is determined by the accuracy of translation of the user's query into the underlying database language (usually SQL) query. If the query translation is accurate, and the underlying database is searched completely, the recall will be one.

For agents like XSAR that search XML repositories, the recall is dependent on the accuracy of translation of the user query into the underlying XML query language, and the existence of a path from the specified starting point to all the XML pages that exist in the repository. For a well-connected repository where every page is reachable from a starting node, the recall will be one. From this discussion, we can see that in general, the recall for the free text methodology would be lower than for the other two methodologies.
Precision

For the free text methodology, precision is determined by a) the features offered to the user to specify a query to a desired level of precision (e.g., concatenated strings, wild card characters, boolean operators), b) the extent to which precision can be sacrificed for response time, and c) the extent of information captured in the underlying search engine database about the real pages. All else being equal, a search engine that offers more options for searching should have greater precision, as compared to one where response time is secondary to precision. For c), a search engine that captures only the title of each page will have less precision than one that captures the title and the META tags of each page. As the number of pages increases, all else being equal, the precision drops.

When using the database schema methodology, the precision of an information repository that uses an underlying database depends on a) the features provided by the user query interface and b) the maximum precision allowed by the underlying query language of the database. If a relational database is used, then the underlying query language is SQL, which provides excellent precision.

If an agent like XSAR is utilized, the query interface fully supports XQL. Thus, the precision of XSAR is identical to the precision of XQL, which supports XPath (http://www.w3c.org), the W3C standard for precision.

Response Time:

The response time in the free text methodology depends on: a) the algorithm used to perform the key word match with the database that has information on each page in the repository and b) the extent to which the designers are willing to sacrifice precision for response time.

When using the database schema methodology, the response time depends on the complexity of the specified query and the extent to which the database is optimized for that query. Thus, for a relational database, a range query on non-indexed attributes with several joins is likely to take significantly more time than a simple SELECT query. Thus, these repositories would need a database administrator to keep track of most frequent queries, and ensure proper tuning access optimizations.

For agents that search XML repositories, response time is determined by the time taken by the agent to crawl through the target repository. This depends on exogenous factors such as network speed, performance of the web server of the target repository, and the size of the target repository. In designing XSAR, we used three endogenous strategies to minimize this time, for given values of exogenous factors: a) a multi-threaded agent, b) a proxy server for caching, and c) providing the user the ability to specify response time threshold. We next describe each of these strategies.

As the agent program fetches new pages from the target repository, it spawns threads to parse these fetched pages. HTML pages are only parsed for links, while XML pages are parsed and searched for the query. One design trade-off here is that spawning a new thread is expensive if the pages being parsed and/or searched are small, while increasing the number of threads pays off in the size of each page in the repository increases.

Finally, XSAR allows the user to set the threshold response time and maximum depth to which the agent should search the target repository. This allows XSAR to be used by different classes of users, for example, one class of users may want to search large repositories comprehensively and leave the agent running, say, overnight, while another class of users may want quicker responses for searches where the maximum depth of the search is set to a finite number.

We tested the performance of XSAR using three experiments described in Appendix 1. In our experiments, a multi-threaded program had the maximum impact in reducing response time, with a proxy cache playing a secondary role. The results (see Appendix 1) indicate that XSAR is clearly slower than the other mechanisms available for searching, which is to be expected, since it searches a target repository dynamically. XSAR trades off faster response time for the ability to search a changing information repository in real time.
Figure 5 summarizes the performance comparison between search mechanisms for the three different methodologies.

Ease Of System Maintenance

In the free text methodology, the major maintenance issues include the upkeep of the document repository, the search engine and the coordination software required to maintain consistency between the internal pages within the organization, and the pages in the information repository. In the database schema methodology, there is a need to track changes to the underlying database schema. Whenever these changes to the schema occur, appropriate changes are needed in the application code that directly queries the database. Furthermore, the organizations have to reveal their schemas to the team maintaining the application code in the system used for information sharing. With XML document repositories, there is an added layer of transparency, since each organization promises to deliver information in the form of XML pages conforming to the DTDs. This allows each organization to hide their internal database schemas and to make schema changes independent of the information sharing process, as long as each organization delivers its required pages. The maintenance issues for the IAIS information sharing system include refreshing pages from each organization and maintenance of the search agent (such as XSAR).

From the above discussion, we can see that in general the database schema method requires the most maintenance, while the free text and IAIS methods require less maintenance.

The results of our comparison of these three methodologies are shown in Figure 6.

From Figure 6, it is clear that the main advantages of using XML documents over the free text methodology are a) the ability to share structured information and b) the ability to search a repository dynamically with greater precision and recall. The main advantages of IAIS over the database schema methodology are a) the former is in general easier to maintain and precludes the need for making database schemas available across organizations, because of the increased layer of indirection provided by DTDs and b) the ability to share unstructured as well as structured information in IAIS. The tradeoff is that queries on XML document repositories require more response time than in the other methodologies.

CONCLUSION AND FUTURE RESEARCH

This chapter proposed a comprehensive framework consisting of three different data integration options in an SOA context, to facilitate sharing of information among government agencies. Information in this domain tends to be both structured and unstructured. This article analyzed the pros and cons of each of the three options, so as to enable decision makers to make better choices when developing inter-agency information sharing plans.

Recently, there has been significantly increased activity in deploying XML in the

*Figure 5. Comparison of methods for information access*

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Recall</th>
<th>Precision</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Text Method</td>
<td>Low</td>
<td>Low</td>
<td>Fast</td>
</tr>
<tr>
<td>Database Schema Method</td>
<td>100%</td>
<td>High</td>
<td>Fast</td>
</tr>
<tr>
<td>XML repositories</td>
<td>100%</td>
<td>High</td>
<td>Slow</td>
</tr>
</tbody>
</table>

Copyright © 2007, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
government. The portal Web site accessible at http://xml.gov, indicates various XML registries that are in progress. The Global Justice XML Data Model (Global JXDM), accessible at http://www.IACPtechnology.org, is one example of interagency cooperation between the US Department of Justice, US Attorney, and the American Association of Motor Vehicle Administrators. In this dictionary, common data definitions are used to support information sharing among prosecutors, law enforcement, courts, and related private and public bodies. As another example, the enactment of the Health Insurance Portability and Accountability Act (HIPAA) of 1996 has also led to a strong motivation to develop a healthcare XML standard.

A new area of research that is emerging is the automatic generation of glossaries from government free text repositories, leading to an ontology layer, as discussed in this article. For example, as pointed out in (Reed, 2003) recent research has resulted in a crawler that sifts through government Web pages that are in free text format and attempts to construct a glossary. Once the glossary is built, the pages are parsed into a large common representation format in XML.

For future work, we plan on a) prototyping a tool to facilitate the information definition aspect by allowing different parties to collaborate on producing XML DTDs and b) validating the usefulness of our framework in real world case studies of inter-agency information sharing, and c) exploring ways to incorporate automated conflict resolution into our framework, to assist agencies in the critical phase of information definition.

REFERENCES


Seminar Conflicts In Geographic And Non-Geographic Databases. Ninth Workshop On Information Technologies And Systems (WITS), Dallas, TX.


ENDNOTES

1. Xerces is available at xml.apache.org, the GMD-IPSI XQI engine is available at: xml.darmstadt.gmd.de/xq/ and tomcat is available at jakarta.apache.org.

2. While alternatives to DTDs such as XML schemas (www.xml.org) do exist, this article focuses on XML DTDs since they are the predominant standard for structuring data on the World Wide Web.

3. This chapter assumes a given size of the universe of pages.

APPENDIX A.

Performance Results of Test Cases

This article presents the results of three test cases that illustrate the performance of XSAR.

The first case was a repository created inside the same network as XSAR. Therefore, the target Web site and the agent server were connected by Ethernet, 10Mbps. The contents of the repository were small HTML files and XML files, less than 4Kbytes each.

The second case was a real XML repository (http://www.xml-cml.org/), which has the conclusive resource for Chemical Markup Language which is used to exchange chemical information and data via the Internet.
The third case was also a real-world case: the Dublin Core Metadata site (http://purl.org/dc/) which is a specification for describing library materials.

Table A1-1 shows the search time for the cases mentioned. The search time results clearly depend on exogenous variables like network conditions, PC hardware specifications, and load status. In order to eliminate at least some variation in these factors, we tested each case five times, and present the mean values in Table 1.

<table>
<thead>
<tr>
<th>Local experimental Environment</th>
<th>CML</th>
<th>Dublin Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number / Average size of HTML</td>
<td>16 / 458 [bytes]</td>
<td>1 / 10490 [bytes]</td>
</tr>
<tr>
<td>Number / Average size of XML</td>
<td>16 / 2756 [bytes]</td>
<td>171 / 10314 [bytes]</td>
</tr>
<tr>
<td>Direct access</td>
<td>254 [ms]</td>
<td>19291 [ms]</td>
</tr>
<tr>
<td>Via cache server</td>
<td>1772 [ms]</td>
<td>14577 [ms]</td>
</tr>
<tr>
<td>Ratio of (w/ Cache) / (w/o Cache)</td>
<td>0.696</td>
<td>0.756</td>
</tr>
</tbody>
</table>

Akhilesh Bajaj is associate professor of MIS, at the University of Tulsa. He received a B Tech in chemical engineering from the Indian Institute of Technology, Bombay in 1989, an MBA from Cornell University in 1991, and a PhD in MIS (minor in computer science) from the University of Arizona in 1997. Dr. Bajaj’s research deals with the construction and testing of tools and methodologies that facilitate the construction of large organizational systems, as well as studying the decision models of the actual consumers of these information systems. He has published articles in several academic journals such as Management Science, IEEE Transactions on Knowledge and Data Engineering, Information Systems and the Journal of the Association of Information Systems. He is on the editorial board of several journals in the MIS area. His research has been funded by the department of defense (DOD). He teaches graduate courses on basic and advanced database systems, management of information systems, and enterprise wide systems.

Suha Ram is Eller professor, management information systems in the College of Business and Public Administration at the University of Arizona. She received a BS degree in mathematics, physics and chemistry from the University of Madras in 1979, PGDM from the Indian Institute of Management, Calcutta in 1981, and a PhD from the University of Illinois at Urbana-Champaign, in 1985. Dr. Ram has published articles in such journals as Communications of the ACM, IEEE Expert, IEEE Transactions on Knowledge and Data Engineering, ISR, Information Systems, Information Science, and Management Science. Dr. Ram’s research deals with modeling and analysis of database and knowledge based systems for manufacturing, scientific and business applications. Her research has been funded by IBM, NCR, US ARMY, NIST, NSF, NASA, NIH, and ORDI(CIA). Specifically, her research deals with interoperability among heterogeneous database systems, semantic modeling, data allocation, schema and view integration, intelligent agents for data management, and tools for database design. Dr. Ram serves on various editorial boards including Information Systems Research, Journal of Database Management, Information Systems Frontiers, and Journal of Systems and Software. She is also the director of the Advanced Database Research Group based at the University of Arizona.