Adaptive Web-based Database Communities

Athman Bouguettaya[1]
Boualem Benatallah[2]
Brahim Medjahed[1]
Mourad Ouzzani[1]
Lily Hendra[3]

[1] Department of Computer Science,
Virginia Tech, USA
{athman, brahim, mourad}@nvc.cs.vt.edu

[2] School of Computer Science and Engineering,
The University of New South Wales, Australia
boualem@cse.unsw.edu.au

[3] Litton PRC, USA
hendra_lily@prc.com

INTRODUCTION

As a result of the rapidly growing number of organizations conducting business over the Web, a large number of heterogeneous information sources (e.g., home pages, tabular data, online digital libraries) is now readily available. The ability to efficiently and effectively share data on the Web is a critical step towards the development of the so-called information super-highway. Existing organizations would form online alliances to deliver integrated value-added information sources (e.g., e-catalogs, information portals).

The evolution into the global information infrastructure and the concomitant increase in the available information, is offering a powerful distribution vehicle for organizations that need to coordinate the use of multiple information sources. However, the technology to organize, search, integrate, and evolve these sources has not kept pace with the rapid growth of the available information space. The efficient sharing of Web data is especially challenging in environments where the information sources are largely autonomous and evolve dynamically. One of the key issues encountered frequently in large cooperative environments, such as data-intensive Web applications, is how users can efficiently query large, intricate, heterogeneous information sources.

Traditional techniques in multidatabases focused on data sharing among a small number of heterogeneous databases (Kim, W. and Seo, J., 1991). Emerging techniques for querying data over the Web focused on information discovery and brokering in the context of unstructured or semi-structured Web-resident data (Florescu, D., Levy, A., and Mendelzon, A., 1998). Our research aims at building a viable infrastructure for integrating and querying Web-accessible databases (Bouguettaya, A., Benatallah, B., Hendra, L., Ouzzani, M., and Beard, J., 2000).

In this chapter, we present our work in the *WebFINDIT* project.

WebFINDIT aims to achieve the scalable integration and efficient querying of Web-accessible databases through the incremental data-driven discovery and formation of interrelationships between information sources. In particular, we present the salient features that are related to the following issues:

- **Information space organization***:* In Web applications, the information space is large and dynamic. On top of that, existing Web tools give little support for the logical organization of data. Thus, the effective use of data in the *anarchic* Web has become enormously complex. *WebFINDIT* uses an *ontological* organization of the information space to filter interactions and accelerate service searches. More precisely, the information space is organized as domain-specific groups. Each group forms a *database community* (also called *ontology*) to represent the domain of interest (some portion of the information space) of the related databases. Databases join and leave communities at their discretion.

- **Queries over database communities***:* An important issue to tackle is how users can efficiently query the potentially vast amount of available information sources. A fundamental premise of our approach is that, in a dynamic environment such as the Web, users would have to be incrementally made aware of available information. Users must be *educated* about the information of interest and thus able to learn on the fly what different databases contain to eventually establish a link to those databases of some interest.

- **Dynamics***:* Database communities operate in a highly dynamic environment. New information sources could come on-line, existing information sources might be removed, etc. Therefore, a key issue is the design of an architecture to cater for dynamic relationships among information communities, as well as relationships among Web-accessible databases and

communities. *WebFINDIT* provides a monitoring mechanism to dynamically alter relationships between different database communities. This is achieved by using distributed *agents* that work as background processes. Their role is to continually gather and evaluate information about the inter-community relationships to recommend changes.

A *WebFINDIT* prototype has been fully implemented in the context of a healthcare application and a working demo is available online (http://www.nvc.cs.vt.edu/~project). Users can navigate in an object graph representing the information source clustering and invoke operations dynamically on these objects.

The remainder of this chapter is organized as follows. In Section 2, we discuss related work. In Section 3, we present the *WebFINDIT*'s approach for information space organization and modeling. In Section 4, we present a language for advertising and querying database communities. In section 5, we present agent support for monitoring and maintaining inter-community relationships. We present the system implementation in Section 6. In Section 7, we give some concluding remarks.

RELATED WORK

In this section, we briefly review some research efforts most related to our work: *information discovery* and *data integration*.

Information Discovery

Information discovery systems, such as *GIOSS* (Tomasic, A., Gravano, L., Lue, C., Schwarz, P., and Haas, L., 1997b) and *Harvest* (Bowman, C., Danzig, P., Schwartz, U. M. M., Hardy, D., and Wessels, D., 1995), focus on building efficient indexing schemes for accessing networked document databases. They rely on keyword or topic-based content indexing techniques (Gudivada, V., Raghavan, V., Grosky, W., and Kasanagottu, R., 1997). In these

systems, resources are typically text documents. *Web search engines* (e.g., *Lycos, Yahoo, Altavista*) and *metasearch tools* (e.g., *MetaCrawler, IBM's InfoMarket*) are examples of information discovery systems (Dreilinger, D. and Howe, A., 1997). The major limitations of information discovery systems are (Konopnicki, D. and Shmueli, O., 1995):

- They do not support structured queries (e.g., SQL-like queries). Query capabilities are limited to content-based expressions (e.g., keywords, sentences, combination of keywords) where returned results usually lack precision.
- They lack abstractions (e.g., database schemas, ontologies) for describing the semantics and organization of information sources. This makes any effective use of available information sources enormously complex.

SQL-like languages for the Web feature support for content and topology-based queries. Examples of such languages include *W3QL* (Konopnicki, D. and Shmueli, O., 1995) and *WebSQL* (Mendelzon, A. O., Mihaila, G. A., and Milo, T., 1996). They use graph-based models to represent Web accessible documents. Other efforts, such as *ARANEUS* (Atzeni, P., Mecca, G., and Merialdo, P.,1997) and *STRUDEL* (Fernandez, M., Florescu, D., Kang, J., Levy, A., and Suciu, D. , 1998), investigated the generalization of these languages to support structured queries. The underlying models provide constructs to describe both the inner structure of Web documents and their hyperlinks. However, these languages provide little support for organizing, integrating, and querying large numbers of dynamic, heterogeneous, and distributed information sources.

Data Integration

Interoperability among loosely coupled and tightly coupled systems has been an active research over the past 20 years. In traditional data integration approaches, the development of an integrated

schema requires the understanding of both the structure and semantics of all database schemas (Bouguettaya, A., Benatallah, B., and Elmagarmid, A., 1998). These approaches are usually acceptable when integrating a small number of non-varying numbers of heterogeneous databases (Ozcan, F., Nural, S., Koskal, P., Evrendilek, C., and Dogac, A., 1997). Interoperability in Web environments requires more flexible and scalable solutions. The volume of information is very large, information formats are more diverse (e.g., XML, HTML, tabular data), and information space is highly dynamic and distributed.

There is a large body of relevant literature on extracting, integrating, and querying Web data (Florescu, D., Levy, A., and Mendelzon, A., 1998). Web-accessible information sources can be structured (e.g., relational database), semi-structured (e.g., XML and HTML documents), or unstructured (e.g., text files). Survey reports on Web information integration approaches can be found in (Kashyap, V., 1997; Florescu, D., Levy, A., and Mendelzon, A., 1998; Bouguettaya, A., Benatallah, B., and Elmagarmid, A. 1998). Current Web data integration approaches propose some interesting capabilities in data extraction (e.g., semiautomatic generation of wrappers), translation, and mediation. An example is the *Object Exchange Model (OEM)* proposed in the *TSIMMIS* project (Garcia-Molina, H. et al., 1997). OEM is a self-describing data model where data can be parsed without reference to any external schema.

Another example of extensions to multidatabase techniques is the flexible query processing technique proposed in the *DISCO* project (Tomasic, A., Amouroux, R., Bonnet, P., Kapitskaia, O., N., H., and Raschid, L., 1997a). DISCO provides support for unavailable information sources and transparent addition of new information sources. Query evaluation takes into consideration partial answers. Other efforts in this area, that include *InfoSleuth* (Bayardo, R. et al. 1997) and *OBSERVER* (Mena, E., Kashyap, V., Illarramendi, A., and Sheth, A., 1998), use agents and ontologies. They feature the use of pre-existing domain specific ontologies to define the terms in each data source. Users formulate their queries

using terms of a selected ontology (local user ontology).

ORGANIZING AND MODELING WEB-ACCESSIBLE DATABASES

In a highly dynamic and large network of databases accessible via the Web, there is a need for a meaningful organization and segmentation of the information space. In *WebFINDIT*, the information space is partitioned into domain-specific communities. Databases join and leave a given community based on their needs and domains of interest. In this section, we present the core concepts of the *WebFINDIT* approach: *database communities, community relationships,* and *database registration*.

Database Communities

Each *database community* is specialized in a single area of interest. It provides domain-specific information (e.g., domain keywords and domain attributes) for interacting within the community and its underlying databases. Database communities provide means for an ontological organization of Web-accessible databases. Such an organization aims at reducing the overhead of discovering databases over the Web. Our approach breaks away from conventional multidatabase systems assumption that database schemas must be *a priori* integrated into a global schema. We argue that the first step in querying Web databases ought to be the discovery of relevant databases. Thus, an appropriate segmentation of the information space has the advantage that the number of potential interactions is restricted.

Figure 1 illustrates the clustering of databases into communities for a healthcare application. It shows sixteen databases grouped into four communities (`Research`, `Medical`, `Insurance`, and `Superannuation`). For example, all databases that contain insurance-related data are members of the `Insurance`

community. The same database may belong to several communities if it deals with different domains of interest. For example, the `Qld Cancer Fund` database is member of `Medical` and `Research` communities.

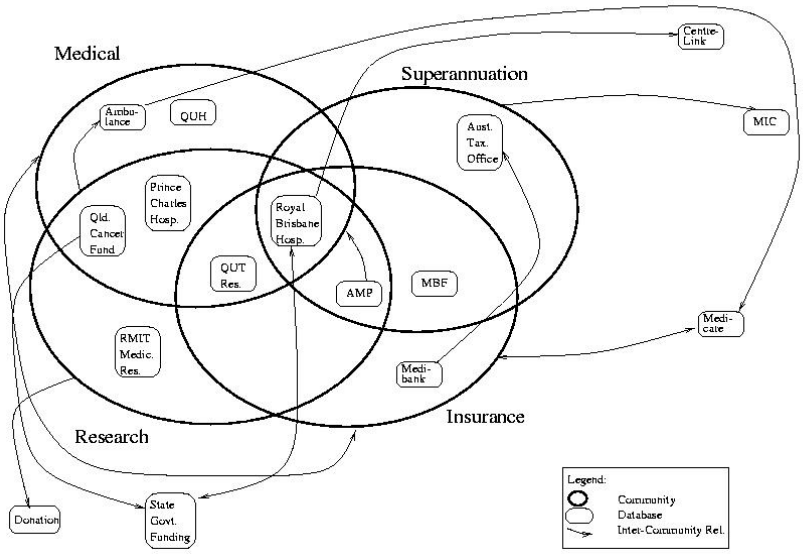

Figure 1. Database Communities

The definition of a community includes the *domain name*, *synonyms*, and *attributes*. The *domain name* is a string that represents the identifier of the community. *Synonyms* provide alternative names of each domain. They can be used to locate databases that provide information about the associated domain. *Domain attributes* constitute a *schema* that can be used to query the members (i.e., databases) of the community. It should be noted that the domain attributes are described without referring to local database schemas. For example, part of the definition of the `Insurance` community is:

Community `Insurance` {
Domain Insurance;
Synonyms {Health Coverage, Social Security,
Medical Expenses};
Schema Insurance_Schema;

}

A subset of the attributes of the schema *Insurance_Schema* (i.e., domain attributes) is:

Community Schema Insurance_Schema {
Attribute String Coverage_Type;
Attribute String Restrictions;
Attribute Double Maximum_Amount;

}

Community Relationships

Databases within a community are organized using a specialization relationship. For example, `Medical Research` is a sub-community of `Research`. A community may have several sub-communities but at most one super-community. Thus, a database community is represented as a tree. The nodes forming the community tree support each other in answering queries forwarded to them. If a query scope conforms to the domain of a given sub-community, then the query will be forwarded to it.

Communities are not isolated entities: they can be related to each other by *inter-community relationships*. *Inter-community relationships* may be viewed as a simplified way to share information with low overhead. They constitute the resources that are available to a community to answer requests that cannot be handled locally. They also represent domain proximity relationships between database communities. Inter-community relationships are used to provide a peer-to-peer topology for connecting communities with overlapping domains. This topology

ensures that if a community cannot process a given request, the request is forwarded to a linked community for possible resolution.

In addition to the domain name, synonyms, and attributes, the definition of a community includes *Sub-Communities* and *Inter-Community Relationships* elements. The following example shows the description of these elements in the definition of the community `Medical Research`.

Community `Medical Research` {
  Super-community `Research`;
  Inter-Community Relationships {`Medical`};
}

The community `Medical Research` is a sub-community of the community `Research`. It also has an inter-community relationship with the community `Medical` (Figure 1).

Database Registration

To become accessible from a given community, a database needs to be registered with that community. The registration of a database consists of providing the following elements:

- *Exported view* of the database schema: It defines attributes that can be used to query the database. It should be noted that different databases might use different local data models (e.g., relational, object-oriented) to describe their local schemas. However, exported views must be defined using the *WebFINDIT* data model (object-based).
- *Wrapper*: It translates *WebFINDIT* queries to local queries expressed using the database native query language. The outputs produced in response to local queries are translated into the format used by *WebFINDIT*.
- *Mapping*: This element allows specifying the mappings between the exported view and community schema. As pointed out, the schema of a community is defined without

directly referring to any database schema. Therefore, when a database is registered with a community, the database provider must define a mapping between the exported view and community schema. We call this mapping *database-community mapping*. The database-community mapping is used to translate a query that is expressed using a community schema into a set of queries spanning individual databases.

• *Documentation*: It provides a human-understandable summary description of the content and capabilities of a database. Documentation may also be associated with a community.

In our healthcare application, `Royal Brisbane Hospital` (`RBH` in short) is a member of the community `Research`. The registration of `RBH` with `Research` is as follows:

Register Database RBH {
Community `Research`;
Exported View RBH_View;
Wrapper "http://www.nvc.cs.vt.edu/WWD-QLOracle";
Database-community Mapping RBHtoResearch;
Documentation "http://www.nvc.cs.vt.edu/RBHDoc";
}

The exported view *RBH_View* and mapping *RBHtoResearch* are:

Exported View RBH_View {
Attribute String ProjectDescription;
Attribute Double AllowedAmount;
}

```
Database-community  Mapping RBHtoResearch {
        Source RBH;
        Target Research;
        Attribute String ProjectDescription IS
                        Research.Topic;
        Attribute Double AllowedAmount IS
                        Research.Funding;
}
```

In this example, the URL "http://www.nvc.cs.vt.edu/RBHDoc" contains documentation about RBH database. The URL "http://www.nvc.cs.vt.edu/WWD-QLOracle" contains the wrapper needed to access RBH. The exported view *RBHView* defines two attributes: *ProjectDescription* and *AllowedAmount*. The mapping *RBHtoResearch* specifies that the attribute *ProjectDescription* (respectively, *AllowedAmount*) in *RBHView* corresponds to `Topic` (respectively, `Funding`) in the schema of the community `Research`.

ADVERTISING AND QUERYING DATABASE COMMUNITIES

An important issue to consider in the Web, is how to efficiently query the potentially vast amount of available databases. For this purpose, we advertise databases and communities in *meta-data repositories*. We also define a declarative language, called the *World Wide Database Query Language (WWD-QL)*, for querying information communities and their databases.

Meta-data repositories

Meta-data repositories contain information that describes the meaning, domain, content, capabilities, and location of databases. To avoid the problem of centralized administration, meta-data

repositories are distributed over the information network. Each database has a *meta-data repository* attached to it. A *meta-data repository* is an object-oriented database that stores information about its associated database and related communities. The schema of the meta-data repository contains a specific class, called *WebDatabase*, which describes general information about the associated database including the location, wrapper, documentation, local query language, and local DBMS. The meta-data repository contains also information about the communities that the database is member of. A subset of the attributes of the class *WebDatabase* is:

Class WebDatabase {
Attribute Set (Community) Communities;
Attribute URL Location;
Attribute URL Documentation;
Attribute URL Wrapper;
Attribute String DBMS;
Attribute String Query-language;
}

The attribute Communities contains references to objects that represent communities of which the database is member. A subset of the attributes of the class Community is:

Class Community {
Attribute String Domain;
Attribute Set (String) Synonyms;
Attribute Community Super-community;
Attribute Set (WebDatabase) Members;
}

Querying Database Communities

The WWD-QL language uses meta-data to locate, browse, and query communities and their underlying databases. This language

combines SQL-like and information retrieval boolean constructs. WWD-QL is designed to query (meta) data over the Web. WWD-QL differs from traditional query languages in that it operates in a large and highly dynamic network of heterogeneous databases. Since the unit of information sharing is the type, this query language is able to query the system at two levels:

- **Meta-data level***:* Queries in this category allow the exploration of the available information space, database location, etc. WWDL-QL provides primitives for educating users about the available information space, locating communities and databases based on constraints over their meta-data. Examples of queries include, *get* communities that are relevant to the topic `Research`, *get* sub-communities of the community `Research`, *get* communities which overlap with the community `Research`, *get* members of the community `Medical Research`, *display* documentation about a specific community or database, and *get* schema of a specific community or exported view of a database.

- **Data level**: After locating relevant communities and understanding their content, users may be interested in querying data stored in the underlying databases. Users may use a community to express queries that require extracting and combining data from multiple members (i.e., databases). We refer to this type of queries as *global queries*. A global query is expressed using a community schema. It is translated into a set of queries to relevant members using the associated database-community mappings and wrappers. Users have also the options to query databases directly. In this case, the query is either expressed using the exported view of the database or embedded in the native query language of the database (e.g., SQL).

MONITORING AND MAINTAINING INTER-COMMUNITY RELATIONSHIPS

The dynamic evolution of inter-community relationships is facilitated by means of *community monitoring agents*. *Agents* are software components characterized mainly by their autonomy and adaptiveness (Nwana, H. S. and Ndumu, D. T., 1997). These characteristics are of prime importance in allowing adaptive inter-community relationships in *WebFINDIT*. Each database community has a monitoring agent associated to it. Monitoring agents contain operational knowledge such as inter-community relationships, their usage statistics, and control policies related to the evolution of inter-community relationships.

Creating and deleting inter-community relationships

We now discuss the creation and deletion of inter-community relationships. The creation of a new inter-community relationship may be beneficial to avoid navigation through communities that are constantly used as bridges between other communities. The deletion of an existing inter-community relationship may be beneficial to reduce the number of *stale links*. Stale links are inter-community relationships that have no useful purpose.

Creating inter-community relationships. Figure 2 illustrates a scenario where a new inter-community relationship is created. In this scenario, the community `Research` has an outgoing inter-community relationship with `Medical`, which in turn has outgoing inter-community relationship with `Insurance`. Assume that during the execution of the system, the monitoring agents of the previous communities report the following: The majority of users who start their query session from `Research` and traverse the inter-community relationship between `Research` and `Medical`, do not initiate queries on the community `Medical`. Rather they use `Medical` as a bridge to go to `Insurance` where

they initiate their queries. In this case, the monitoring agent of `Research` would recommend the creation of a new inter-community relationship from `Research` to `Insurance`. This would allow users to navigate directly from `Research` to `Insurance` and reduce the number of traversed nodes to reach relevant communities.

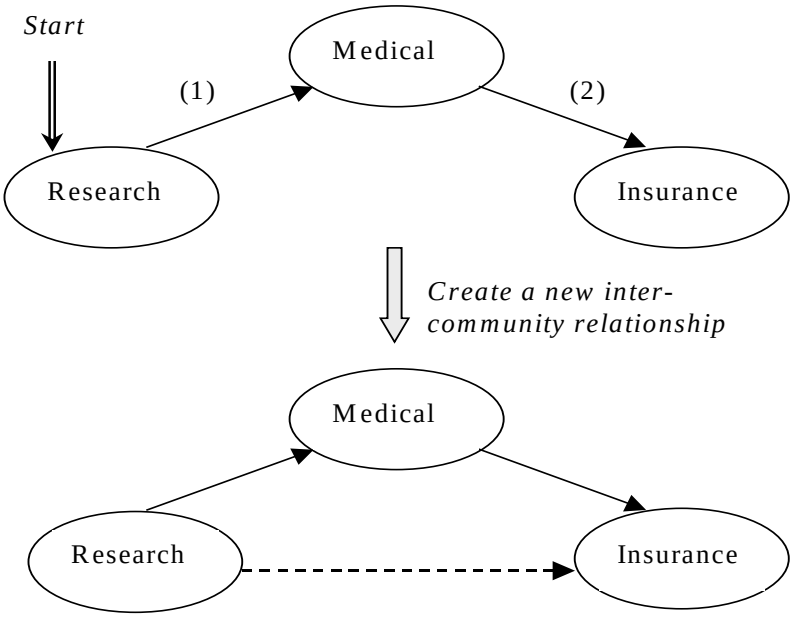

Figure 2. Creation of an Inter-community Relationship

Deleting inter-community relationships. If an inter-community relationship between two database communities is rarely used or always leads to a non-relevant community, then it is most likely to be stale. The related agent would recommend its deletion. Consider the example of Figure 3. The community `Medical` has an outgoing inter-community relationship with the community `Insurance` which in turn has an outgoing inter-community relationship with the community `Superannuation`. `Medical` has another outgoing inter-community relationship with `Superannuation`. Assume that the monitoring agents of the

previous communities report the following: The majority of users who navigate directly from `Medical` to `Superannuation` ultimately leave `Superannuation` without performing any query. This may suggest that the direct link between `Medical` and `Superannuation` does not seem to be too beneficial. In this case, the monitoring agent of `Medical` would recommend the deletion of the inter-community relationship between `Medical` and `Superannuation`.

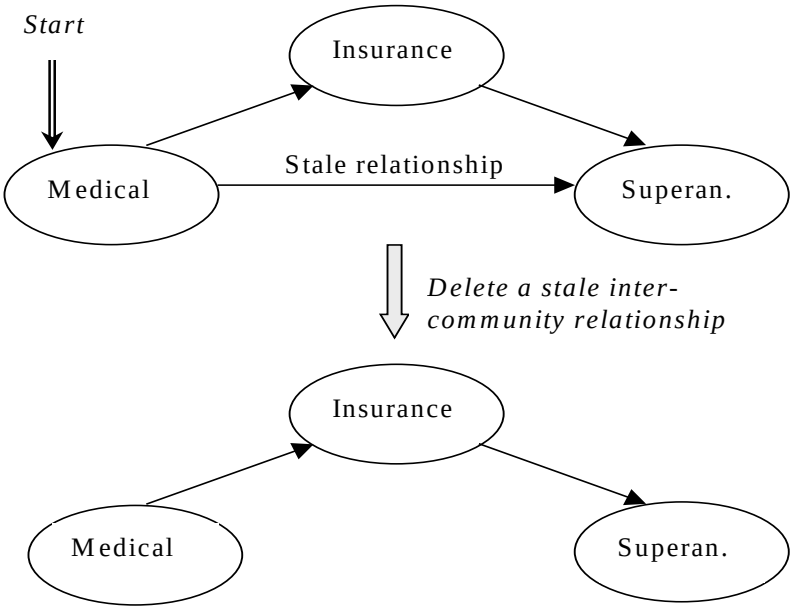

Figure 3. Deletion of an Inter-community Relationship

Monitoring Inter-community Relationships Navigation

Community monitoring agents primarily keep track of user's navigation over the different inter-community relationships. Each inter-community relationship will be characterized by a number of parameters (e.g., number of times an inter-community relationship is traversed) continually gathered by the associated agent. Based

on these parameters, monitoring agents may recommend the *creation* or *deletion* of inter-community relationships. In a nutshell, community agents monitor traffic over inter-community relationships and record final destinations of navigation sessions through these relationships. Each agent gathers and stores statistics about all outgoing and incoming inter-community relationships of its associated community. It may also query remote agents about the statistics they gathered so far.

Inter-community relationship navigation statistics. For each inter-community relationship, the associated monitoring agent gathers the following statistics: (i) The number of times the relationship has been traversed, (ii) The number of times the relationship leads to a final destination, and (iii) the number of times the relationship leads to a non-final destination.

A community is considered as a *final* destination of an inter-community relationship if users submit queries (i.e., global or local queries) to that community. It is considered as a *non-final* destination when users do not submit queries to it. In this case, users follow other inter-community relationships, backtrack to the source community (if there is an inverse inter-community relationship), or leave the system. An agent can also query other agents about final destinations navigation through a given inter-community relationship.

Change recommendations. Agents can be queried by community administrators to get information about inter-community relationships usage. They are also programmed to periodically notify information about inter-community relationships usage. The results of queries to a monitoring agent may consist of recommendations to create or delete inter-community relationships.

A monitoring agent reports that an inter-community relationship is stale (and hence recommends its deletion) if the value of a specific parameter, called *DP* (*Deletion Parameter*), is less than a certain threshold (e.g., 20%). The threshold is

predefined by community owners or administrators. *DP* is obtained using the following ratio:

- *DP= Number of times an inter-community relationship led to a final destination/Number of times this relationship was traversed.*

A monitoring agent will confirm that a new inter-community relationship is worth creating if the value of a specific parameter, called *CP* (*Creation Parameter*), is greater than a certain threshold (e.g., 80%). *CP* is computed using the following formulae:

*CP = A * B* where:

- *A = Number of times an inter-community relationship was a non-final destination / Number of times this relationship was traversed.*
- *B* is the *DP* of the final destination. It is obtained by querying the agent associated with that destination.

If the value of *CP* is greater than a pre-defined threshold, then the agent recommends the creation of a new inter-community relationship between the starting community and the final destination.

WebFINDIT IMPLEMENTATION

This section presents the overall architecture of *WebFINDIT*. This architecture adopts a client-server approach to provide services for interconnecting a large number of distributed, autonomous and heterogeneous databases. At the communication layer *WebFINDIT* integrates major middleware technologies such as *CORBA*, *DCOM*, *EJB*, and *RMI*. Database gateways such as *JDBC* and *ODBC* are used to access databases. Monitoring agents are implemented using *Voyager 2.0*, an agent-enhanced *Object Request Broker* (*ORB*). *Voyager* supports agent mobility. It is among the very few

agent platforms that support full native *CORBA IDL*, *IIOP* and bi-directional *IDL/Java* conversions.

Architecture

The *WebFINDIT* components are grouped into four layers (Figure 4): *query layer*, *communication layer*, *meta-data layer*, and *data layer*.

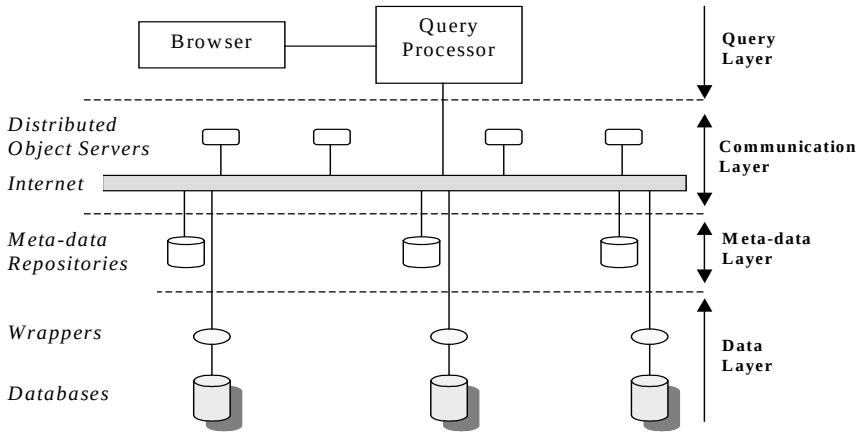

Figure 4. WebFINDIT Layers

Query layer. This layer provides users access to *WebFINDIT* services. It allows users to browse, search, and access communities and databases using graphical and text queries. This layer contains two components: user interface and query processor.

Communication Layer. This layer manages the interaction between *WebFINDIT* components. It mediates requests between the query processor and meta-data/database servers. This component is implemented using a network of communication middleware including *CORBA*, *EJB*, *DCOM*, and *RMI*.

Meta-data Layer. This layer consists of a set of meta-data

repositories. Each repository stores meta-data about the associated databases (i.e., locations, wrappers, communities, etc). Meta-data is stored in *ObjectStore* databases.

Data Layer. This layer has two components: databases and wrappers. The current version of *WebFINDIT* supports relational (*mSQL*, *Oracle*, *Sybase*, *DB2*, and *Informix*) and object oriented databases (*ObjectStore*). A wrapper provides access to a specific database server.

Prototype

Figure 5 depicts details of the *WebFINDIT* architecture using a healthcare application. As a proof of concept, seventeen databases along with their respective meta-data repositories (total of 34 databases) have been used. Host operating systems are *Unix* (*Sun Solaris*) and *Windows NT*. Different distributed object middleware have been used to interconnect databases: three *CORBA ORBs* (*Visibroker*, *Orbix*, and *OrbixWeb*), two Sun *RMI* servers, one *WebLogic EJB* server, and one *Microsoft DCOM* server.

Access to *WebFINDIT*'s databases is handled by the *query processor*. The query processor provides access to databases via *JDBC* for relational databases in *Unix* platforms, *ODBC* for databases on *NT* machines, and *C++* method invocations for object-oriented databases. All meta-data repositories are implemented using *ObjectStore*. The use of an object-oriented database was dictated by the hierarchical structure of the schema associated to meta-data repositories. We used four *CORBA Orbix ORBs* to represent the existing communities. A fifth *ORB* was added for meta-data repositories associated to databases that do not belong to any community. Each meta-data repository is registered to a given *ORB* through a *CORBA* object wrapper.

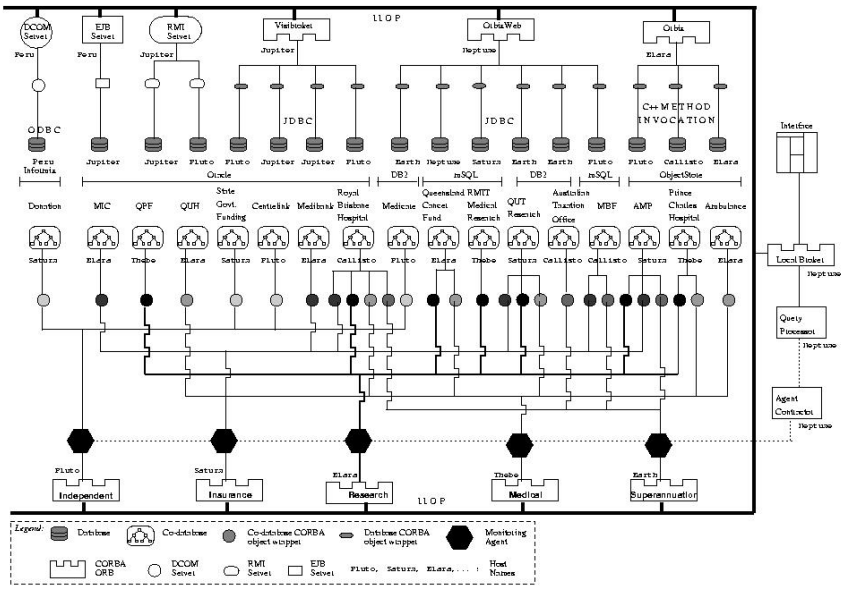

Figure 5. Detailed WebFINDIT Architecture.

The *agent contractor* informs monitoring agents (implemented in *Voyager 2.0*) when users move from one community to another. One monitoring agent is associated to each database community. It stores information about destinations of all outgoing and incoming inter-community relationships. This information makes it possible for the agents to determine the usefulness of inter-community relationships. Database administrators are presented with the results obtained by community agents.

A sketch of the algorithm executed by monitoring agents is presented in the following. The function *BuildStatistics* is executed by each agent in collaboration with the contractor. Monitoring agents update their statistics (stored in *mSQL* databases) under the control of the agent contractor, since it is the only component that knows when users traverse an inter-community relationship. The function *EvaluateStatistics* is executed by relevant agents. The *timeframes* (weekly, monthly, half-yearly, yearly) determine the

frequency to perform evaluation on each inter-community relationship. Timestamps are used to determine whether a timeframe has elapsed.

```
Function BuildStatistics /* Executed by each agent, with help from
                            the agent contractor */
Begin
  - Record the number of traversals for each inter-community
    relationship
  - Determine whether a destination community is the final/non-
    final destination as far as the relationship is concerned
End
```

```
Function EvaluateStatistics /* Executed by each agent */
Begin
    From the first to the last entry in the statistics
    Do
        - Look at the timeframes chosen by the user, if any of them
          has elapsed
        - Determine whether the entry is a stale relationship
        - Determine whether the entry produces a new relationship
          If the entry produces a stale or new relationship
              Then present the agent's recommendation to the
                   system administrator
End
```

CONCLUSION

WebFINDIT provides a framework that fosters Web data integration in a scalable and adaptive way. It proposes an incremental and self-documenting approach to share Web data. The system processes users' queries in two steps: (i) querying meta-data for communities location and semantic exploration, and (ii) querying selected databases for actual data. Since scalability

and flexibility are of great importance in Web-based environments, our integration framework features appropriate abstractions. First, the community-like organization and segmentation of the information space in meaningful subspaces makes database search and sharing more efficient. The descriptions of communities allow querying Web-accessible databases without knowing their schemas. Second, inter-community relationships are established to allow query migration among communities. If a query cannot be resolved in the local community, it is forwarded to relevant community through an inter-community relationship. Third, inter-community relationships are dynamically maintained by monitoring agents. Monitoring agents collect statistics about relationship usage to suggest the creation or removal of inter-community relationships. A working prototype is fully operational and available online (http://www.nvc.cs.vt.edu/~project).

We are currently investigating the use of *WebFINDIT* to access *digital government* databases (Bouguettaya, A., Ouzzani, M., Medjahed, B., and Cameron, J., 2001). We are collaborating with the *Family and Social Services Administration* (*FSSA*), a government agency providing social services to needy families and citizens. We are also leveraging *WebFINDIT* to provide seamless integration of *Web Services* (i.e., modular applications that are programmatically accessible via the Web) and data into one uniform architecture (Benatallah, B., Medjahed, B., Bouguettaya, A., Elmagarmid, A., and Beard, J., 2000). Another on-going work is to use XML (*eXtensible Markup Langugage*), the *de facto* standard for data representation and exchange on the Web, to support meta-data repositories.

Acknowledgment: The first author's work was in part supported by an NSF grant number 9983249-EIA.

REFERENCES

Atzeni, P., Mecca, G., and Merialdo, P. (1997). Semistructured and Structured Data in the Web: Going Back and Forth. ACM SIGMOD, 26(4).

Bayardo, R. et al. (1997). InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA.

Benatallah, B., Medjahed, B., Bouguettaya, A., Elmagarmid, A., and Beard, J. (2000). Composing and Maintaining Web-based Virtual Enterprises. VLDB Workshop on Technologies for Web Services, Cairo, Egypt.

Bouguettaya, A., Benatallah, B., and Elmagarmid, A. (1998). Interconnecting Heterogeneous Information Systems. Kluwer Academic Publishers.

Bouguettaya, A., Benatallah, B., Hendra, L., Ouzzani, M., and Beard, J. (2000). Supporting Dynamic Interactions among Web-based Information Sources. IEEE Transactions on Knowledge and Data Engineering, 12(5).

Bouguettaya, A., Ouzzani, M., Medjahed, B., and Cameron, J. (2001). Managing Government Databases. IEEE Computer, 34(2).

Bowman, C., Danzig, P., Schwartz, U. M. M., Hardy, D., and Wessels, D. (1995). Harvest: A Scalable, Customizable Discovery and Access System. Technical Report, University of Colorado, Boulder, USA.

Dreilinger, D. and Howe, A. (1997). Experiences with Selecting Search Engines Using Metasearch. ACM Transactions on Information Systems, 15(3).

Fernandez, M., Florescu, D., Kang, J., Levy, A., and Suciu, D. (1998). Catching the Boat with Strudel: Experience with a Web-site

Management System. ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA.

Florescu, D., Levy, A., and Mendelzon, A. (1998). Database Techniques for the World-Wide Web: A Survey. ACM SIGMOD Record, 27(3).

Garcia-Molina, H. et al. (1997). The TSIMMIS Approach to Mediation: Data Models and Languages. Journal of Intelligent Information Systems, 8(2).

Gudivada, V., Raghavan, V., Grosky, W., and Kasanagottu, R. (1997). Information Retrieval on the World Wide Web. IEEE Internet Computing, 1(5).

Kashyap, V. (1997). Information Brokering over Heterogeneous Digital Data: A Metadata-based Approach. PhD thesis, New Brunswick, The State University of New Jersey, USA.

Kim, W. and Seo, J. (1991). Classifying Schematic and Data Heterogeneity in Multi-base Systems. IEEE computer, 24(12).

Konopnicki, D. and Shmueli, O. (1995). W3QS: A Query System for the World Wide Web. International Conference on Very Large Data Bases, Zurich, Switzerland.

Mena, E., Kashyap, V., Illarramendi, A., and Sheth, A. (1998). Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. International Conference on Formal Ontologies in Information Systems, Trento, Italy.

Mendelzon, A. O., Mihaila, G. A., and Milo, T. (1996). Querying the World Wide Web. International Conference on Parallel and Distributed Information Systems, Florida, USA.

Nwana, H. S. and Ndumu, D. T. (1997). An Introduction to Agent Technology. Lecture Notes in Artificial Intelligence, Springer Verlag, 1198.

Ozcan, F., Nural, S., Koskal, P., Evrendilek, C., and Dogac, A. (1997). Dynamic Query Optimization in Multidatabases. IEEE Data Engineering, 20(3).

Tomasic, A., Amouroux, R., Bonnet, P., Kapitskaia, O., N., H., and Raschid, L. (1997a). The Distributed Information Search Component (Disco) and the World Wide Web. SIGMOD Record, 26(2).

Tomasic, A., Gravano, L., Lue, C., Schwarz, P., and Haas, L. (1997b). Data Structures for Efficient Broker Implementation. ACM Transactions on Information Systems, 15(3).