Collaboratively Sharing Scientific Data

Fusheng Wang[1] and Cristobal Vergara-Niedermayr[2,⋆]

[1] Integrated Data Systems Department, Siemens Corporate Research, Princeton, New Jersey, USA
fusheng.wang@siemens.com
[2] Freie Universität, Berlin, Germany
vergara@mi.fu-berlin.de

Abstract. Scientific research becomes increasingly reliant on multi-disciplinary, multi-institutional collaboration through sharing experimental data. Indeed, data sharing is mandatory by government research agencies such as NIH. The major hurdles for data sharing come from: i) the lack of data sharing infrastructure to make data sharing convenient for users; ii) users' fear of losing control of their data; iii) difficulty on sharing schemas and incompatible data from sharing partners; and iv) inconsistent data under schema evolution. In this paper, we develop a collaborative data sharing system *SciPort*, to support consistency preserved data sharing among multiple distributed organizations. The system first provides Central Server based lightweight data integration architecture, so data and schemas can be conveniently shared across multiple organizations. Through distributed schema management, schema sharing and evolution is made possible, while data consistency is maintained and data compatibility is enforced. With this data sharing system, distributed sites can now consistently share their research data and their associated schemas with much convenience and flexibility. SciPort has been successfully used for data sharing in biomedical research, clinical trials and large scale research collaboration.

Keywords: Scientific Data Sharing, Scientific Data Integration, Biomedical Data Management, Computer Supported Collaborative Work, Schema Sharing, Schema Evolution.

1 Introduction

With increased complexity of scientific problems, scientific research is increasingly a collaborative effort across multiple institutions and disciplines. Scientific researchers need an effective infrastructure to share their complex data, results, and the experiment settings that generate the results. Therefore, researchers are able to reuse experiments, pool expertise and validate approaches. As stated in the NIH roadmap and blueprint initiatives [1], to achieve the need to develop new partnerships of research, we need to represent and record clinical research

⋆ Work done while visiting Siemens Corporate Research.

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 805–823, 2009.

information, exchange and share such information through standard information protocol, provide a modern information technology platform.

For example, NIH provides large-scale collaborative project awards for a team of independently funded investigators to synergize and integrate their efforts, and the awards mandate the research results and data to be shared [2,3]. As another example, Siemens Medical Solutions has research collaborations with hundreds of research sites distributed across the US, each providing Siemens marketing support by periodically delivering white papers, case reports, clinic methods, clinic protocols, state-of-the-art images, etc. In the past, data were delivered through media such as emails, CDs and hard copies. As a result, deliverable content was non-centralized, therefore difficult to manage, integrate, search, and reliably archive.

Besides, clinical trials are often distributed among multiple hospitals or medical research institutes. For example, University of California, Irvine is leading a group of universities to conduct Diffuse Optical Spectroscopy based clinical trials. Patients are recruited at distributed institutions and experiments are performed on these patients. These require a platform to collect both clinical data and experiment data at multiple distributed institutions, and integrate and share them together for patient study and data analysis.

While the user demands for scientific data sharing have only increased with time, it is still difficult for researchers to find data sharing solutions to support their collaborative research. Given the strong demand, the lack of viable solutions can be attributed to the following reasons. i) The lack of data sharing infrastructure for convenient data sharing. Cyberinfrastructures such as Grid based systems (CaBIG [4], Biomedical Informatics Research Network (BIRN) [5]) focus on large scale data sources and are heavy weight for regular research sites; ii) Users' fear of losing control of their data. Researchers would rather have maximal control of their data on a server located on their own labs, instead of "outsourcing" them somewhere else. Each Principle Investigator and its collaborators will naturally form a research unit and produces a data source. Even when data are shared, the researchers may still want to have flexible sharing control – they keep the ownership of the data, and can revoke the sharing of the data at any time. iii) Difficulty on sharing data under the same or compatible schemas and incompatible data formats from sharing partners. Each site may use different schemas and represent data in different formats; and iv) Inconsistent data under schema evolution. As schemas can keep evolving as applications change, data consistency can be broken at either the same site or across multiple sites.

To meet these demands and solve the problems, we develop SciPort [6], a scientific data management and sharing system for collaborative scientific research. SciPort brings together the following essential components to support scientific data sharing:

- Generic XML based scientific data management which provides a unified data model to represent scientific data (Section 2);
- Lightweight and fully controlled data sharing architecture through a Central Server (Section 3);

- Flexible schema sharing for multiple distributed research sites (Section 4);
- Distributed uniform schema management to keep data and schema consistent across multiple sites (Section 5);
- Distributed multiform schema management to provide flexible schema evolutions at each Local Server while maintaining data compatibility on Central Servers (Section 6); and
- a schema change detection tool to detect and visualize changes between schemas to support schema management (Section 7).

2 Overview of Scientific Data Management with SciPort

SciPort was first developed as a scientific data management system for scientific data modeling, data authoring, management, viewing, searching and exchange. SciPort takes an XML based approach for data modeling, schema representation, and storage and queries.

2.1 Scientific Data Modeling

Scientific experiments often consist of complex steps or processes. A complex scientific experiment can be modeled as a set of atomic objects. The context information of such objects can then be represented through a unified document model, *SciPort Document*. SciPort Document can represent both (nested) structured data, files and images.

A SciPort document includes several objects: i) Primitive Data Types/Fields. Primitive data types are used to represent structural data, including *integer*, *float*, *date*, *text*, and Web-based data types such as *textarea*, *radiobutton*, *checkbox*, *URL*, etc.; ii) File. Files can be linked to a document through the file object; iii) *Reference.* A reference type links to another SciPort document; iv) Group. A group is similar to a table, which aggregates a collection of fields or nested groups. There can be multiple instances for a group, like rows of a table; and v) Category. A category relates a list of fields, e.g., "patient data" category, "experiment data" category, etc. Categories are used only at the top level of the content, and categories are not nested (Figure 1).

2.2 XML Based Implementation

The model can be best implemented as XML. The hierarchical nature of the data model fits very well with the tree based XML data model. Users can also easily define their own schemas which are internally represented with an XML-based schema definition language. Here we take the native XML database approach to manage scientific documents, where standard XML query language XQuery [7] is supported. We provide two options: Oracle Berkeley DB XML [8], an open source embedded XML database, and IBM DB2 with pureXML^{TM} support. SciPort provides comprehensive Web-based tools to support authoring and searching [6].

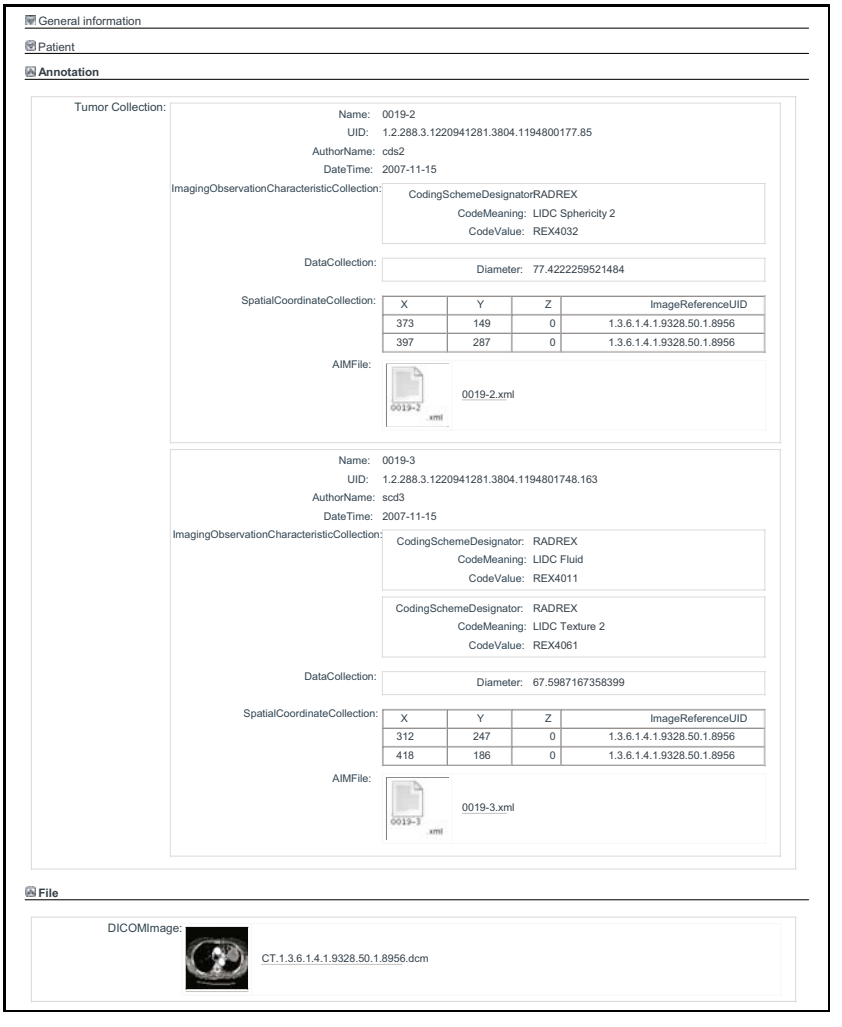


Fig. 1. A Sample SciPort Document

3 Sharing Distributed Scientific Data

As scientific research increasingly becomes a collaboration activity, researchers frequently need to collaborate through sharing their data. There are several common requirements for the data sharing: i) Convenience: data sharing should be a single step action; ii) Data ownership: researchers own and have full control of their data; iii) Flexible Sharing Control: while data can be shared, data sharing can also be revoked by researchers at any time; iv) Up-to-date of shared data. As data are updated or removed, corresponding shared data also need to be synchronized accordingly to stay current.

3.1 Sharing Data through a Central Server

To meet the above requirements and support closer collaboration and integration, we develop a distributed architecture to share and integrate data through a Central Server (Figure 2). In this architecture, each research site will have their own Local Server which itself functions as an independent Server for data collection, management, search, and report. In addition, there will be an additional Central Server upon which Local Servers are able to selectively publish their data (structured documents) (Figure 3). Images/files, which are often the major source of data volume, are still stored on corresponding Local Servers but are linked from the published documents on the Central Server. Once a user on the Central Server tries to download a document from the Central Server, actual data files are downloaded from the corresponding Local Server that holds the data.

Thus, the Central Server provides a global view of shared data across all distributed sites, and can also be used as a hub for sharing schemas among multiple sites. Since data are shared through the metadata (SciPort Documents), the integration is lightweight. Users on the Central Server will only have read access to the data.

Figure 2 illustrates an example SciPort sharing architecture formed by four Local Servers at four universities: UCI, UCSF, Dartmouth and Penn. Each Local Server is used for data collection and management of clinical trial data at its local institution. Since these clinical trials are under the same research consortia,

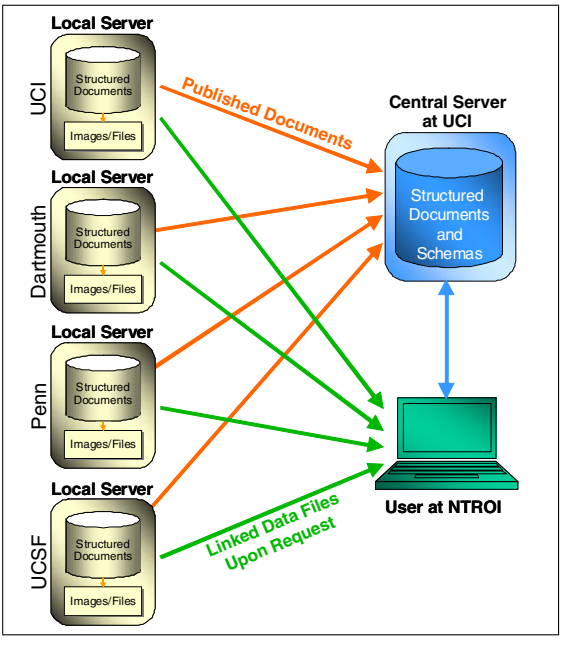


Fig. 2. The Central Server Based Architecture for Data Sharing

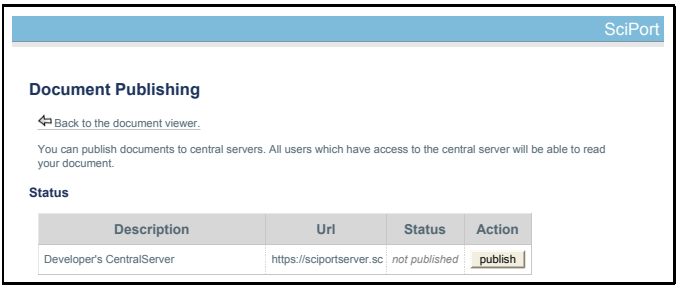


Fig. 3. An Example of Publishing an Existing Document

they would share their data together by publishing their data (documents) to the Central Server located at UCI. Members at NTROI research consortia are granted read access on such shared data through the Central Server. Once the user identifies a data set from the Central Server and wants to download it, the user will be redirected to the corresponding Local Server that hosts the data to download the data to the client.

3.2 Data Synchronization

One requirement for the data sharing is to keep shared data up-to-date. The following operations are related to document synchronization.

- Create. When a document is created, the author or publisher has the option to publish this document (Figure 4). Once the document is published, a "published" status is added to the document. A user can also set up an automatic publishing flag so all new documents will be automatically published;
- Update. When an update is performed on a published document, the document will automatically be republished to the Central Server;
- Delete. When a published document is deleted, it will also be automatically removed from the Central Server;
- Unpublish. A user can also stop sharing a document by unpublishing the document.

3.3 Security and Trust between Local Servers and the Central Server

Server Verification. The trust between Local Servers and the Central Server is implemented through security tokens. For a Local Server to be accepted into the network, it will be granted a security token to access the Central SciPort Server services. The token will be imported at the setup step. When a Local Server tries to connect to the Central Server, the Central Server verifies if the token matches.

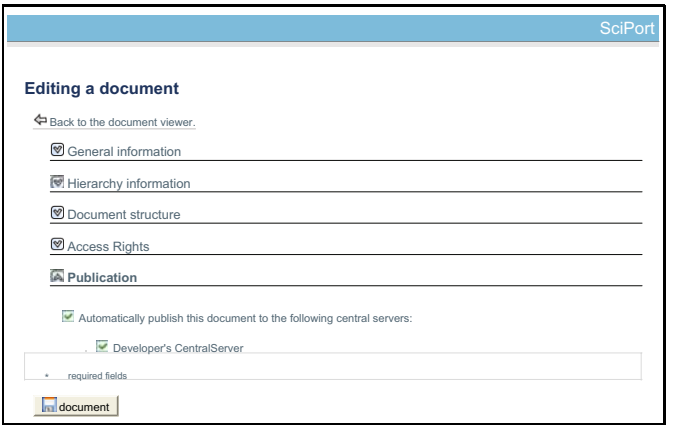


Fig. 4. An Example of Publishing a New Document

Single Sign-on and Security. One issue for sharing data from distributed databases is that it is not feasible for Central Server users to login to every distributed database. Since once a user publishes a document, the user grants the read access of the document (including the files attached to the document) to the users on the Central Server, thus it is unnecessary for another authentication. Therefore, users on the Central Server should be able to automatically access shared data from a Local Server in a transparent way.

To support this, the Local Server Document Access Control Manager has to make sure that the remote download requests really come from Central Server users who are currently logging on. We develop a single sign-on method to guarantee the security of the data sharing, consisting of the following steps as shown in Figure 5.

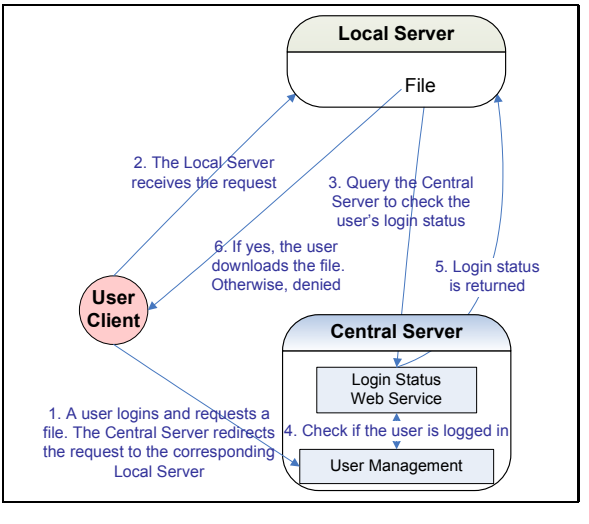


Fig. 5. Single Sign on for Central Server

3.4 Sharing Data in Multiple Data Networks

Data can be shared not only in a single data network through one Central Server, but also in multiple data networks through multiple data networks. One organization may want to share the same data in multiple networks, as demonstrated in an example (Figure 6). There are two networks one centered at UCI, and another one centered at Stanford. One institution UCI is collaborating with both networks and needs to share data with both networks. UCI will be granted as a partner site and its Local Server will be configured for both networks. When a document is being published, the target Central Server can be either of them or both of them. This sharing architecture make it possible for very flexible data sharing.

The Benefits. This data sharing architecture provides many benefits: i) data sharing is as convenient as a single click; ii) users have full control of their data, and can revoke the sharing at any time; iii) shared data on the Central Server always remain updated, and iv) the Central Server based sharing architecture makes it possible to conveniently share schemas, as discussed next.

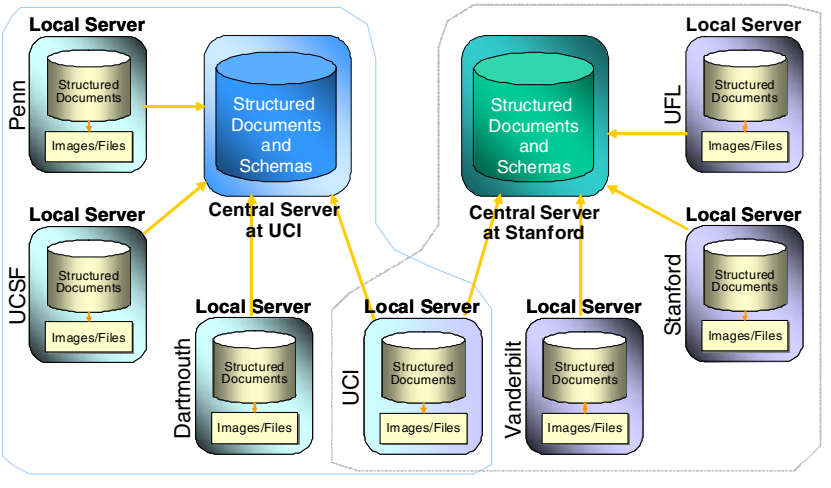


Fig. 6. Sharing Data in Multiple Data Networks

4 Sharing Schemas

Schemas are used to define the data structures and constraints of documents. The former includes a mix of (possibly nested) object types defined in the data model, and the latter includes i) number of instance constraints for file and group types, ii) minimal and maximal value constraints; and iii) controlled values (Figure 7).

Schemas are an essential component since they are used for i) data validation; ii) document authoring form generation; iii) data presentation – templates are defined based on schemas; iv) search form generation. Sharing schemas are critical

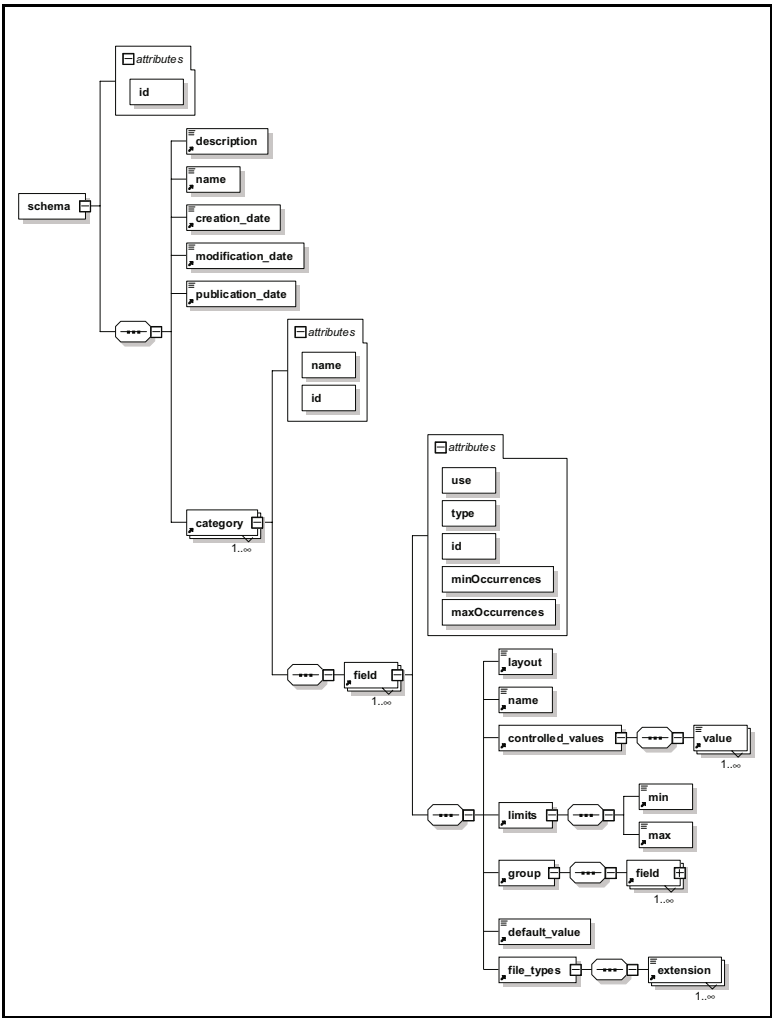


Fig. 7. Schema Model Diagram

for sharing data, since the Central Server glues data together using shared schemas to present and search data. How to keep data from multiple Local Servers coherent is also dependent on at what level and how schemas can be shared.

4.1 Publishing Schemas

Schemas can be shared by publishing them to the Central Server. From a Local Server *Schema Management* menu, schemas created on the Local Server can be selectively published to target Central Servers, as shown in the example in Figure 8. When a new document is being published to the Central Server, the availability of the corresponding schema will be checked. If not, the schema will also be published.

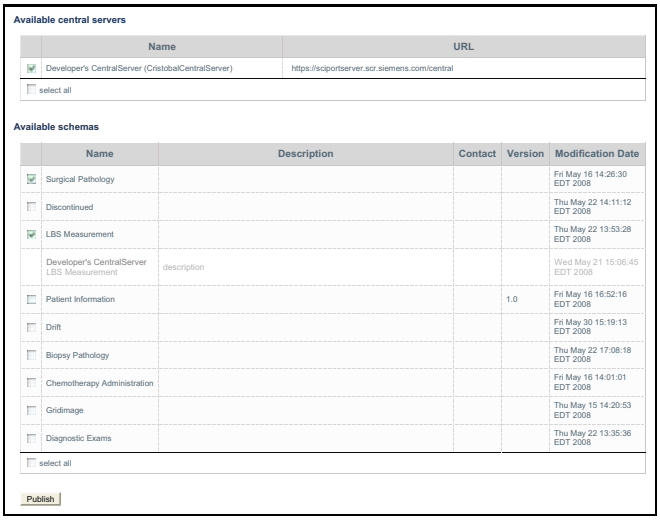


Fig. 8. An Example of Publishing Schemas from a Local Server

The *owner of a schema* is defined as the Local Server on which the schema is first created. A schema is identified by its owner and schema ID.

SciPort also provides comprehensive access control management, and two roles are related to schema management: i) *organizer* role with privileges to authoring and update schemas and ii) *publisher* role with privileges to publish documents and schemas.

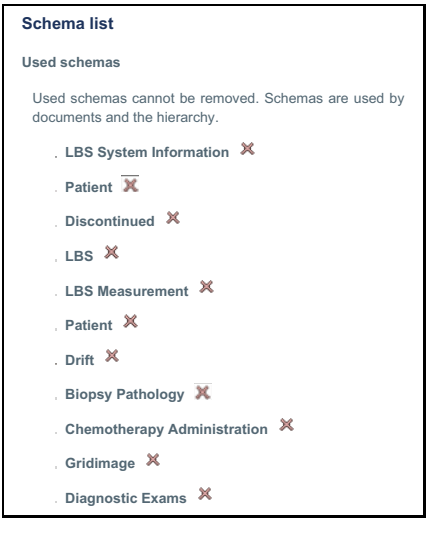
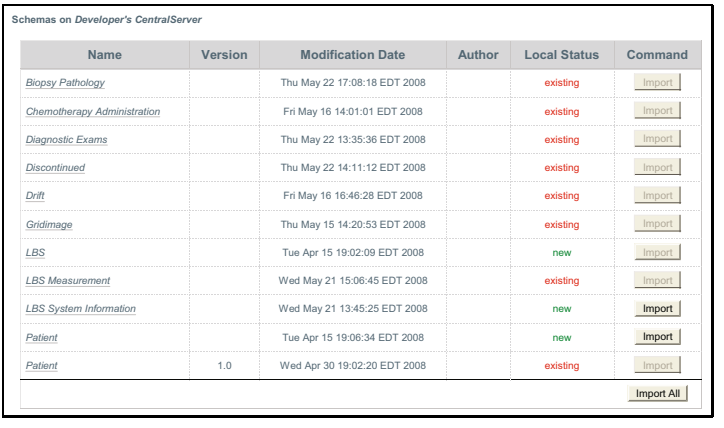


Fig. 9. An Example of Schemas Shared on a Central Server

Once a schema is published, it can be shared through the Central Server. Figure 9 shows a sample list of schemas published from Local Servers. Schemas can also be removed from the Central Server to stop the sharing.

Other Local Servers can reuse schemas by importing schemas from the Central Server, as shown in Figure 10.


Schemas on *Developer's CentralServer*

Name	Version	Modification Date	Author	Local Status	Command
Biopsy Pathology		Thu May 22 17:08:18 EDT 2008		existing	Import
Chemotherapy Administration		Fri May 16 14:01:01 EDT 2008		existing	Import
Diagnostic Exams		Thu May 22 13:35:36 EDT 2008		existing	Import
Discontinued		Thu May 22 14:11:12 EDT 2008		existing	Import
Drift		Fri May 16 16:46:28 EDT 2008		existing	Import
Gridimage		Thu May 15 14:20:53 EDT 2008		existing	Import
LBS		Tue Apr 15 19:02:09 EDT 2008		new	Import
LBS Measurement		Wed May 21 15:06:45 EDT 2008		existing	Import
LBS System Information		Wed May 21 13:45:25 EDT 2008		new	Import
Patient		Tue Apr 15 19:06:34 EDT 2008		new	Import
Patient	1.0	Wed Apr 30 19:02:20 EDT 2008		existing	Import

Import All

Fig. 10. An Example of Importing Schemas to a Local Server

Unpublish. A schema can be unpublished from a Central Server by a Local Server, thus the schema is not available on the Central Server for further sharing.

Remove from Central Server. Users on the Central Server with an "organizer" role can remove a schema from the Central Server if no document on Central Server is using this schema. This can be used to cleanup non-used schemas.

4.2 Three Scenarios of Schemas Sharing

Based on the use cases, there are three typical scenarios of schema sharing:

Static Schema. A schema is fixed and will not be changed. For example, some common standard based schemas are not likely to change. Once a schema is authored and changed to the fixed status, it can be published to the Central Server to be shared by every Local Servers (Figure 11). This is the simplest scenario, as schemas can be created once and shared directly through the Central Server.

Uniform Evolving Schema. A Schema can be changed and a uniform version is shared by all Local Servers, thus data consistency is maintained across all sites. The schema is owned by its original creating Local Server, and the owner can make certain changes.

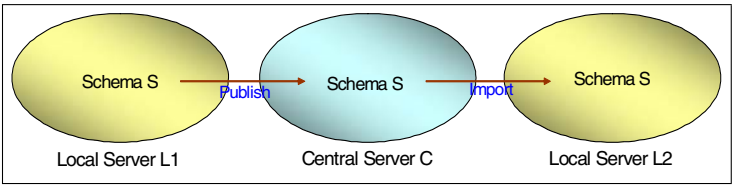


Fig. 11. Static Schemas Sharing

Multiform Evolving Schema. A "seed" schema is first created and shared as public – every Local Server becomes the owner and can update the schema. The Central Server maintains a version of the schema that conditionally merges updates from Local Servers, thus all documents published on the Central Server will be compatible under this schema.

Next, we will discuss how to manage schemas and their evolution for the last two scenarios (static schema management is straightforward and ignored here.)

5 Uniform Schema Sharing and Management

Since schemas can be shared and used across multiple distributed data sources, one challenging issue is how to manage schema evolution while keeping the document consistent and compatible with their schemas in a distributed environment. To solve this problem, we first define the following principles for distributed schema management and sharing:

- Minimal administration: no central management on schemas, and the Central Server serves as an information exchange hub;
- Data consistency: schema evolution has to be backward compatible otherwise the integrity of documents will be broken;
- Control of schemas: only the owner of a schema can update a schema;
- Current of Schema: the Central Server always has the up-to-date version of compatible schema if that schema is shared by the owner;
- Sharing Maximization: Only the last time publisher can unpublish a schema from the Central Server.

On a Local Server, schemas can be created, updated, deleted, imported, and published or unpublished. Schemas can also be removed from the Central Server to stop the sharing. To enable data consistency, each schema operation has to be carefully defined.

5.1 Uniform Schema Management

Next we discuss schema operations and their conditions.

Creation. A schema can be created on a Local Server if the user has the organizer role. This Local Server will become the owner of this schema.

Incompatible Update. Incompatible update is the one that can lead to inconsistency between the new schema and existing documents if any. Incompatible updates are forbidden unless the following conditions are met: i) only the owner of a schema can make updates to the schema; ii) there are no existing documents created using this schema on this Local Server; and iii) the schema was never published, i.e., there will no other Local Servers using the schema to create documents. A publishing status is associated with each schema.

Compatible Update. A user updates an existing schema and such updates will not lead to inconsistency between the new schema and existing documents if any. The following conditions are required for compatible update:

- *Ownership.* The Local Server is the owner of the schema, and the user is the organizer on this Local Server.
- *Field Containment.* All the fields in the last schema are present in the new schema, belong to same category, and belong to the same group if any. There can be new categories and fields added.
- *Type Compatibility.* All the fields in the new schema have the same type or a compatible type, i.e., a more general type. For example, an integer type can be updated to a float type.
- *Relaxed Value Range Constraints.* No new value constraints are permitted for existing fields which do not have any constraint. Value constraints can be updated with more relaxed ranges. i.e., the new maximal limit should be greater than or equal to last one, and the new minimal limit should be less than or equal to last one. Constraints on new fields are permitted.
- *Relaxed Controlled Values.* For field with controlled values, the extent is enlarged with more options. Similarly for radiobuttons and checkboxes, more options are added.
- *Relaxed Constraints on Number of Instances.* For group field or file field, there can be a constraint on the minimal and/or maximal number of instances. No instance number constraint is permitted on existing fields, and instance number constraint can be updated with more relaxed range. i.e., the new maximal limit should be greater than or equal to last one, and the new minimal limit should be less than or equal to last one. Constraints on new fields are permitted.

Change of a field's order within its sibling is not considered incompatible.

Once a schema is updated on the Local Server, it will be automatically republished to the Central Server (if any) onto which the schema has been published. This will ensure the Central Server always maintains up-to-date versions of schemas.

Delete. A schema can be deleted if the user has the organizer role, and there are no documents using this schema on this Local Server. If the Local Server is the owner of the schema, and the schema is never published, deleting a schema will

eliminate the schema forever. If the schema was once published, it may be still alive and used on other servers.

Publish. A schema can be published to one or multiple Central Servers if the user has a publisher role and there is no newer version of this schema on the Central Server. There are three scenarios of schema publishing:

- Schemas can be manually published from the schema publishing interface;
- The schema of a document is automatically published when a document is published. When a document is published, the Local Server will check the Central Server if the schema is available or up-to-date. Otherwise the schema is republished;
- If a schema was published and is then updated with compatibility, the new version schema will be automatically republished on the Central Server and replace the last version. This will keep the schemas on the Central Server up-to-date.

Unpublish. A schema can be unpublished by a Local Server with the following conditions: i) the user has the "publish" role on the Local Server; ii) there is no document associated with it on the Central Server, and iii) the Local Server is the last publisher of the schema. The last condition is necessary. Otherwise if the schema is used at multiple Local Servers, every Local Server can easily stop the publishing which can be against the sharing goal of the last publisher.

Import. A Local Server can import a shared schema from the Central Server if the user has the organizer role. When there is a new version of a schema on the Central Server, it will be automatically detected by a Local Server when there is a document being published from that Local Server.

Remove from Central Server. Users on the Central Server with the organizer role can remove a schema from the Central Server if no document on Central Server is using this schema. This can be used to cleanup non-used schemas.

5.2 An Example of Uniform Schema Management

A user with organizer privilege on a Local Server L1 creates a schema S(V1) (Figure 12). The user may later find the schema not accurate, and makes changes to the schema. Since no document has been created and the schema has never

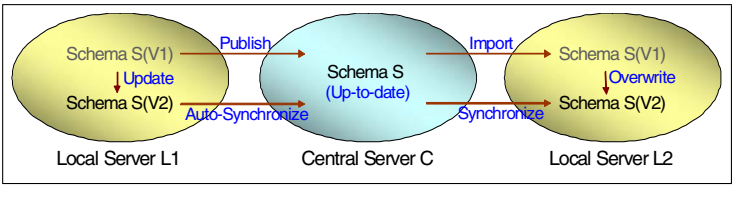


Fig. 12. Uniform Schema Sharing

been shared yet, the user may make arbitrary changes, including incompatible updates. Once the user has a stable usable version of the schema, the user begins to author documents based on this schema, and publishes documents to the Central Server C. Schema S(V1) will be automatically published to the Central Server. After some time, the user may need more information for their data or adjust existing fields, and need to update schema S. Since there are existing documents using the schema, and the schema was also published, the user can make compatible update to schema S(V1). (If compatible update is not sufficient, the user has to create a new schema.) Once schema S(V1) is updated as S(V2), it will be automatically propagated to the Central Server to replace the last version S(V1).

A user with organizer privilege on a Local Server L2 imports schema S(V1) after S(V1) is published on the Central Server, and documents then are create on L2 on this schema. Later S(V2) replaces S(V1) on the Central Server, which is detected when a document of schema S(V1) is published to C. The user at L2 will be prompted to synchronize the schema, and S(V1) is replaced by S(V2) on L2.

6 Distributed Multiform Schema Sharing

Uniform schema sharing provides data compatibility through a single uniform version of schemas across all servers, and maintains the ownership of schemas and provides controlled schema evolution. There can also be cases that multiple sites need to adapt certain schema to their own needs, and a uniform schema may not be feasible. To provide flexible schema evolution at each Local Server and support data compatibility on the Central Server, we provide a multiform schema sharing approach.

As shown in Figure 13, a "seed" schema S will be first created and made as public owned, and then published onto the Central Server. Each Local Server will be able to import this schema, and adjust the schema for its local use (S_{L1}, S_{L2}, etc.) Schema updates from each site will be conditionally merged on the Central Server as S_M. The goal of the merging is to keep the schema version on the Central Server mostly relaxed, through the following merging conditions:

– If the changes are incremental structural changes, they will be merged. These include adding of new fields or new categories;

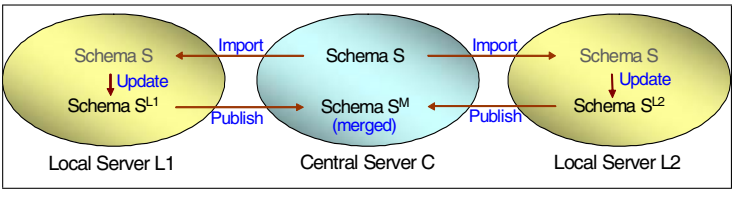


Fig. 13. Multiform Schema Sharing

- If the constraints on the Central Server in the central server are more permissive than the new or updated constraints that the modification suggests, then the suggested changes are ignored by the Central Server;
- Structural removal changes such as field removal or category removal will be ignored and not merged.

In this way, each Local Server will maintain its local schema evolution while the Central Server will maintain a merged schema for shared data compatibility.

7 Schema Change Detection

To support schema management, schema change detection tool is needed to detect changes (structural changes and constraint changes, compatible and incompatible changes) and visualize changes for users. SciPort uses XML based schema representation to represent both complex data structures and constraints, as shown in Figure 7. The schema has two types of objects: category and field. Category is the top level object, and a field can be nested if it is of group type. Each object is represented with an unique ID. We develop a schema change detection tool, thus changes can be easily visualized through XSLT.

The algorithm is shown in Algorithm 1. It compares nodes from the base schema (*NODE_B*) and nodes from the target schema (*NODE_T*). For the child nodes of two comparing nodes, the child nodes are ascendingly sorted based on their IDs (S1 and S2 respectively), thus the comparison will be on two sorted lists. Starting from the root, the algorithm will walk through each node S1, and try to identify target node with the same ID. For nodes with smaller IDs on the target S2, they will be identified as new nodes; for target node with identical ID, for non-group fields, compatibility is checked; for group fields, they are further recursively compared. If identical ID node is not the last one on S2, next node in S1 is picked up and the process is repeated until the end. The changes generated include structural changes (NEW, DELETE), incompatible constraint changes (INCOMPATIBLE_CHANGE), and compatible constraint changes (COMPATIBLE_CHANGE). Changes are ordered based on the schema hierarchy, thus they can be easily visualized through applying an XSLT template.

8 Related Work

With the increasing collaboration of scientific research, collaborative cyberinfrastructures have been researched and developed in the past [9,10]. Grid-based systems(such as caBIG – cancer Biomedical Informatics Grid [4], Biomedical Informatics Research Network (BIRN) [5]) provide infrastructures to integrate existing computing and data resources. They rely on a top down common data structure. This is difficult to get agreement upon and requires much effort and cost to setup and maintain such systems. SciPort is lightweight and can be quickly customized for either research labs or research networks.

Algorithm 1. schema_node_comparison (*NODE_B*, *NODE_T*, *CHANGE*)

```
1: L1 ← List of child nodes of NODE_B
2: L2 ← List of child nodes of NODE_T
3: NEW = NULL
4: DELETE = NULL
5: INCOMPATIBLE_CHANGE = NULL
6: COMPATIBLE_CHANGE = NULL
7: if NODE_B is root then
8:    CHANGE = NULL
9: end if
10: if L1 = NULL then
11:    NEW = L2;
12:    CHANGE += NEW;
13: else if L2 = NULL then
14:    DELETE = L1
15:    CHANGE += DELETE;
16: else
17:    S1 = SORT L1 in ascending order by ID
18:    S2 = SORT L2 in ascending order by ID
19:    P = first node in S2
20:    for N = firstnodeinS1 to N = lastnodeinS1 do
21:       if id(N) = id(P) then
22:          if N is a group then
23:             CHANGE' = NULL
24:             CALL schema_node_comparison (N, P, CHANGE')
25:             CHANGE += CHANGE'
26:          else
27:             if P != N then
28:                INCOMPATIBLE_CHANGE = incompatible changes
29:                COMPATIBLE_CHANGE = compatible changes
30:             end if
31:          end if
32:       else if id(N) < id(P) then
33:          DELETE = F
34:       else if id(N) > id(P) then
35:          NEW = P
36:          while P = P → next and id(N) > id(P) do
37:             NEW += P
38:          end while
39:          P = P → next
40:          if id(N) = id(P) then
41:             if N is a group then
42:                CHANGE' = NULL
43:                CALL schema_node_comparison (N, P, CHANGE')
44:                CHANGE += CHANGE'
45:             else
46:                if P != N then
47:                   INCOMPATIBLE_CHANGE = incompatible changes
48:                   COMPATIBLE_CHANGE = compatible changes
49:                end if
50:             end if
51:          end if
52:       end if
53:       CHANGE += NEW + DELETE + INCOMPATIBLE_CHANGE + COMPATIBLE_CHANGE
54:    end for
55: end if
```

A context-based sharing system is proposed in [11], which focuses on tools instead of data as in SciPort.

While Grid based systems are more used on sharing computing and storage resources, P2P is more used on sharing data [12]. MIRC [13] is a popular pure-P2P based system for authoring and sharing teaching files, which is hard to extend for generic data management. SciPort provides much generality on scientific data management and supports much more powerful integration.

A publish and subscribe architecture for distributed metadata management is discussed in [14], which focuses on the synchronization problems. In [15], an approach of bottom-up collaborative data sharing is proposed, where each group independently manages and extends their data, and the groups compare and reconcile their changes eventually while tolerating disagreement. Our approach takes an approach in between the bottom-up and top-down approaches, where each group manages their data, but also achieves as much agreement on schemas as possible through controlled schema evolution.

Extensive work has been done in data integration and schema integration [16,17]. Our system takes a proactive approach where schema and data consistency is enforced during data authoring and schema authoring.

9 Conclusion

Contemporary scientific research is moving towards multi-disciplinary, multi-institutional collaboration. These lead to strong demand for tools and systems to manage and share the data. This drives the development of SciPort – a Web-based data sharing system for collaborative scientific research. Through a lightweight Central Server based sharing architecture, SciPort provides convenient ways for not only data sharing, but also schema sharing, where data owners still have full control of the data. SciPort further provides flexible approaches to manage schemas and their evolution while maintaining data consistency. Besides static schema sharing, SciPort makes it possible to share a uniform evolving schema across multiple sites, or even site-varied multiform evolving schemas while compatibility is maintained on the sharing server. The system is being well received by research communities, and has been successfully deployed in many research organizations.

References

1. NIH Roadmap Initiatives, `http://nihroadmap.nih.gov/initiatives.asp`
2. NIH Statement on Sharing Scientific Research Data, `http://grants2.nih.gov/grants/guide/notice-files/NOT-OD-03-032.html`
3. Piwowar, H., Becich, M., Bilofsky, H., Crowley, R.: Towards a Data Sharing Culture: Recommendations for Leadership from Academic Health Centers. PLoS medicine 5(9) (September 2008)
4. caBIG: cancer Biomedical Informatics Grid, `http://caBIG.nci.nih.gov/`
5. Biomedical Informatics Research Network, `http://www.nbirn.net/`

6. SciPort Wiki, https://sciportserver.scr.siemens.com/mediawiki
7. XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/
8. Oracle Berkeley DB XML, http://www.oracle.com/database/berkeley-db/xml/
9. Arzberger, P., Finholt, T.A.: Report on Data and Collaboratories in the Biomedical Community Workshop (2002), http://nbcr.sdsc.edu/Collaboratories/CollaboratoryFinal2.doc
10. Revolutionizing Science and Engineering Through Cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure (2003), http://www.communitytechnology.org/nsf_ci_report/report.pdf
11. Chin, Jr.,G., Lansing, C.S.: Capturing and Supporting Contexts for Scientific Data Sharing via the Biological Sciences Collaboratory. In: CSCW (2004)
12. Foster, I., Iamnitchi, A.: On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In: IPTPS 2003 (2003)
13. MIRC, http://mirc.rsna.org
14. Keidl, M., Kreutz, A., Kemper, A., Kossmann, D.: A Publish & Subscribe Architecture for Distributed Metadata Management. In: ICDE (2002)
15. Taylor, N.E., Ives, Z.G.: Reconciling while Tolerating Disagreement in Collaborative Data Sharing. In: SIGMOD (2006)
16. Halevy, A., Rajaraman, A., Ordille, J.: Data integration: the teenage years. In: VLDB (2006)
17. Doan, A., Halevy, A.Y.: Semantic integration research in the database community: A brief survey. AI Magazine 26(1), 83–94 (2005)