frontiers in
NEUROINFORMATICS

ORIGINAL RESEARCH ARTICLE
published: 30 January 2009
doi: 10.3389/neuro.11.002.2009

Distributed XQuery-based integration and visualization of multimodality brain mapping data

Landon T. Detwiler[1], Dan Suciu[2], Joshua D. Franklin[1], Eider B. Moore[1], Andrew V. Poliakov[1], Eunjung S. Lee[3], David P. Corina[4], George A. Ojemann[5] and James F. Brinkley[1,2,3]*

[1] *Department of Biological Structure, University of Washington, Seattle, WA, USA*
[2] *Department of Computer Science and Engineering, University of Washington, Seattle, WA, USA*
[3] *Department of Medical Education and Biomedical Informatics, University of Washington, Seattle, WA, USA*
[4] *Department of Linguistics, University of California at Davis, Davis, CA, USA*
[5] *Department of Neurological Surgery, University of Washington, Seattle, WA, USA*

Edited by:
Maryann E. Martone, University of California, San Diego, USA

Reviewed by:
Jeffrey S. Grethe, University of California, San Diego, USA
Gwen Jacobs, Montana State University, USA
Jonathan Nissanov, Drexel University College of Medicine, USA

****Correspondence:***
James F. Brinkley, Department of Biological Structure, University of Washington, Box No. 357420, Seattle, WA 98195, USA.
e-mail: brinkley@u.washington.edu

This paper addresses the need for relatively small groups of collaborating investigators to integrate distributed and heterogeneous data about the brain. Although various national efforts facilitate large-scale data sharing, these approaches are generally too "heavyweight" for individual or small groups of investigators, with the result that most data sharing among collaborators continues to be *ad hoc*. Our approach to this problem is to create a "lightweight" distributed query architecture, in which data sources are accessible via web services that accept arbitrary query languages but return XML results. A Distributed XQuery Processor (DXQP) accepts distributed XQueries in which subqueries are shipped to the remote data sources to be executed, with the resulting XML integrated by DXQP. A web-based application called DXBrain accesses DXQP, allowing a user to create, save and execute distributed XQueries, and to view the results in various formats including a 3-D brain visualization. Example results are presented using distributed brain mapping data sources obtained in studies of language organization in the brain, but any other XML source could be included. The advantage of this approach is that it is very easy to add and query a new source, the tradeoff being that the user needs to understand XQuery and the schemata of the underlying sources. For small numbers of known sources this burden is not onerous for a knowledgeable user, leading to the conclusion that the system helps to fill the gap between *ad hoc* local methods and large scale but complex national data sharing efforts.

Keywords: data integration, distributed query processing, XQuery, query shipping, brain mapping, neuroinformatics, semantic web, brain visualization

INTRODUCTION

Ongoing improvements in the quality and quantity of available techniques have provided neuroscience researchers with multiple means of observing regions of brain activation in subjects performing behavioral tasks. This proliferation of data gathering techniques has lead to large volumes of data available regarding brain function. Because these data are often distributed, highly heterogeneous and sometimes contradictory, it has also emphasized the need for data organization and integration in order to develop theories about brain function.

Initially under the auspices of the Human Brain Project (HBP) (Koslow and Hyman, 2000), and more recently under other neuroinformatics efforts, there have been many efforts to integrate and share brain mapping information. Many of these efforts have resulted in centralized databases accessible through portals such as the neuroscience gateway (Society for Neuroscience, 2008). Examples of these centralized databases include SenseLab (Miller et al., 2001), SUMS (Van Essen, 2008) LONI (Laboratory of Neuro Imaging, 2008), the fMRI Data Center (Gazzaniga, 2008), and the Cell Centered Database (Martone et al., 2003). The utility of these sites is directly related to the willingness of investigators to take the time to deposit their data since in most cases there is no requirement that data be submitted as a condition of publication (Nature Neuroscience, 2000).

Recognizing that not all investigators will want to or even be able to submit their data to a central site, projects like BIRN (Keator et al., 2008) and caBIG (Oster et al., 2008) have established large scale grid based approaches to data sharing, in which members of the consortium maintain their data at their local sites and make them available to a federated data integration system. The BIRN project is especially relevant because its primary focus is neuroscience.

Although BIRN and caBIG facilitate large-scale data integration and sharing on a national level, a significant effort is involved in integrating local lab data into these networks. Thus, these networks may not be well suited to a small group of collaborating labs who initially only wish to share and integrate data among themselves.

At the current time most data sharing among small groups of labs is done by email or similar *ad hoc* methods. Thus, there is a need for tools to facilitate small-scale integration and sharing of data. However, to be practical such tools need to be "lightweight", not requiring great effort to setup; they must be easy to use by non-programmers, they must allow for semantic interoperability, and they must be scalable both as new data sources are added, and as the sources become included in larger efforts such as BIRN.

To approach this problem we are developing a lightweight distributed query-based architecture, in which any web-service encapsulated data source may be included as long as it accepts incoming queries and returns XML results. Adding such a source is as simple as including it as a subquery in a distributed XQuery (the W3C recommended query language for XML) that is processed by a modified XQuery processor we have developed. The advantage of this approach is that it is very easy to add a new source, although the tradeoff is that a developer (not necessarily end-user) must know XQuery and the schemata of the sources.

The key informatics novelty in our approach is that it is XML-based rather than the more traditional SQL-based approach to distributed query systems. Although many such distributed query systems have been developed for relational databases (Ozsu and Valduriez, 1999), less is known about effective and efficient methods of querying networks of semi-structured data like XML, particularly for large datasets. In fact, other than our own previous work in distributed XML databases (Bales et al., 2005; Re et al., 2004) we are aware of only one other effort in this area (Fernandez et al., 2007).

In the remainder of this paper we describe our lightweight distributed XML-based data integration system, which we call DXBrain. Beginning with a driving neuroscience use case we discuss the overall architecture of the system, the individual data sources, the web service wrappers which process source queries and deliver XML results, the distributed XQuery engine, and the web user application and results visualization. We also illustrate example uses of the system. We conclude by discussing the tradeoffs in this approach versus more heavyweight approaches, the generality and scalability of this approach for other biomedical data sources, and the work that remains to be done in order to make this approach more easily usable by non-programmers. We also point to two downloadable Java libraries that could be useful to others wishing to deploy their own distributed query system.

MATERIALS AND METHODS

The development of DXBrain has been driven by the need to understand language organization in the brain. Under the auspices of the Human Brain Project, the University of Washington (UW) Structural Informatics Group (SIG) has constructed data management, visualization, and analysis systems to aid neuroscience researchers in the study and mapping of language centers in the human brain (Brinkley, 2008; Brinkley et al., 1997).

The language mapping data sources differ in modality (e.g. textual transcriptions of patient responses, 2D operative photographs, 3D MRI volumes), data model (e.g. textual data might be tabular in nature, such as CSV or relational, or it could be hierarchical like XML), data format (e.g. tab delimited vs. comma separated data), and data storage (e.g. flat files vs. relational database systems). Although the data are highly heterogeneous, the sources also share unifying information. In particular many sources contain information about the same set of subjects, often performing the same sets of language tasks (i.e. fMRI images of patients performing a set of object naming tasks vs. textual transcriptions of their responses to the same set of tasks performed while undergoing neurosurgery). These common information elements provide the basis for integration.

ARCHITECTURE

The overall architecture of the DXBrain system consists of the following four layers, reading from the bottom up in **Figure 1**:

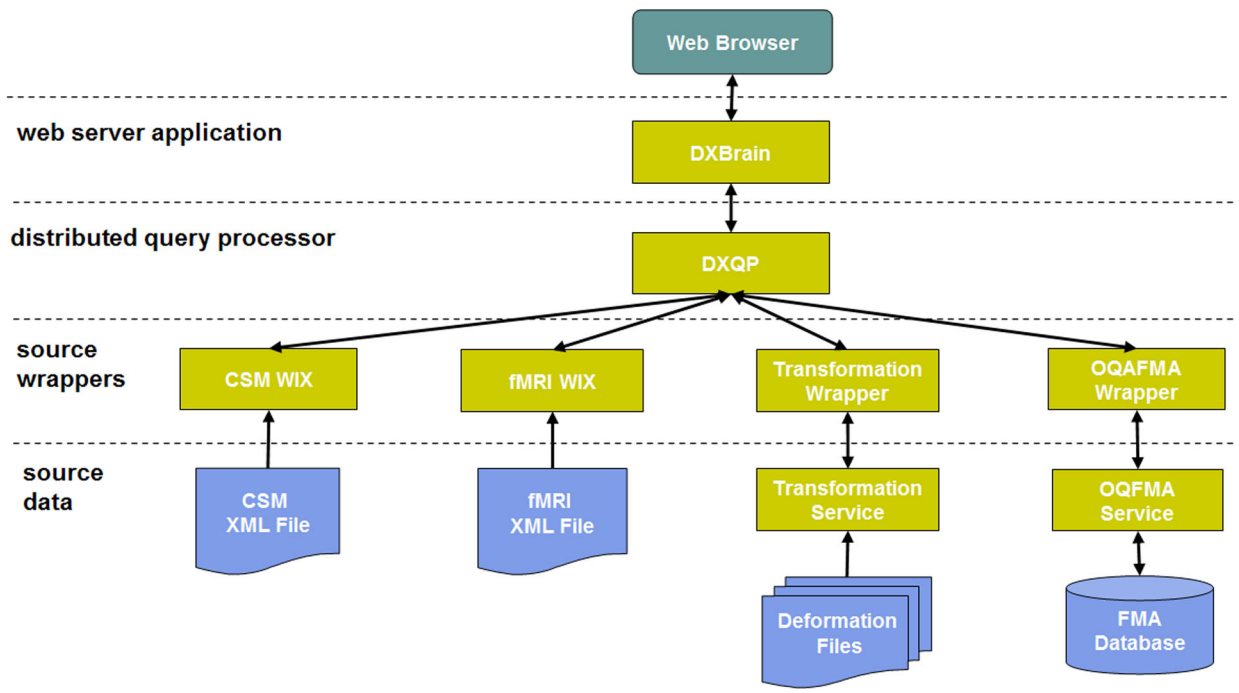

FIGURE 1 | System architecture.

Source data: Data sources are separately maintained at distributed locations on the network. The data models and/or formats of these sources are limited, conceptually, only by the source wrappers at the next level of our architecture. In our current system, we have two types of sources: XML files and application interfaces.

Source wrappers: Data sources are made available for query by means of source wrappers. All wrappers present a query method, accepting source-specific sub-queries, and returning XML results. These wrappers are accessible as web services.

Distributed query processor: The Distributed XQuery Processor (DXQP) takes distributed XQueries (or DXQueries) as input, delegates the appropriate sub-queries to the appropriate source wrappers, gathers all wrapper results, and constructs unified query result documents in XML.

Web server application: The DXBrain application itself accepts queries from the user, communicates with DXQP to evaluate user queries, and provides multiple means for formatting and/or visualizing the query results. It also provides facilities for saving/retrieving/editing user queries.

In the remainder of this section we describe the specific implementation of these layers in greater detail.

Source data and source wrappers

As a data integration system, DXBrain must support the incorporation of multiple data sources. In our driving biological application, as in many other scientific experiments, the data sources are highly heterogeneous in terms of the data model (i.e. relational data model vs. hierarchical data model) and in terms of the data format (i.e. CSV vs. Excel vs. relational tables).

When interacting with our system we wanted these heterogeneous sources to behave in a homogeneous manner. We achieve this by wrapping our sources in a common query interface. It is the job of the wrapper to translate incoming queries into source specific data requests and to transform source results into appropriate XML.

We have developed a Java application called WIX (Web Interface to XML) that can auto-generate an XQuery web-service encapsulation of any valid XML document (regardless of schema) (Structural Informatics Group, 2007b). WIX is essentially a wrapper generator. As such it is very easy to add a new XML data source to the network. Although standard XQuery provides the ability to process arbitrary XML documents through the "doc" command, the "doc" command implements the *document shipping* paradigm, in which the entire document must be shipped to the query processor. WIX, on the other hand, implements the *query shipping* paradigm, in which only the query and query results need be shipped over the network.

The WIX website shows a demo of its use, and provides a downloadable version that can be used to add any new XML file to the DXBrain source network. The program is an executable Java JAR file that takes an arbitrary XML file as input (with optional XML Schema), and generates a Web Archive (WAR) file that can deployed as a web service by dropping it into a container such Apache Tomcat (Apache Software Foundation, 2007). The service then provides a method for evaluating an XQuery over the XML file, and for returning the schema if it exists.

Although many data sources will be in the form of web-service encapsulated XML documents, they do not all need to be. In fact all that is required is that the source provide some sort of web accessible query method that returns XML results. In fact the source does not even need to be a static database, but can instead be an interface to an application that dynamically generates XML results.

In the following sections we illustrate these different kinds of data sources using examples from our driving application. These sections refer to the boxes at the bottom two levels of **Figure 1**.

Cortical stimulation mapping data. The Cortical Stimulation Mapping (CSM) database contains information gathered by neurosurgeons during intra-operative studies of epileptic patients (Brinkley et al., 2004; Hinshaw et al., 2002; Ojemann et al., 1989). The study procedure involves direct electrical stimulation of localized points on the surface of a patient's brain while simultaneously asking the patient to perform object naming tasks (i.e. the patient is shown a picture of a boat and asked to name the object that they see). Object naming errors are correlated with the stimulation site location (in X–Y–Z patient-specific coordinates) in order to identify brain regions essential for language function.

An XML export of the CSM data source is loaded into a WIX web service wrapper (CSM WIX in **Figure 1**) exposing CSM data to the DXBrain system. SilkRoute, an earlier tool we developed to process XQueries over relational data, facilitates this export (Fernandez et al., 2002; Re, 2006).

fMRI data. The fMRI data source stores MRI (magnetic resonance imaging) and fMRI (functional magnetic resonance imaging) data for many of the same patients found in the CSM source (performing the same object naming tasks as in the CSM case). This data source contains image processing metadata produced by our XBatch plugin (Poliakov et al., 2007) to the SPM fMRI analysis package (Wellcome Department of Cognitive Neurology, 2001).

Like the CSM database, an XML export of the fMRI data source is made available to DXBrain via a WIX wrapper (fMRI WIX in **Figure 1**).

Transformation service. This source is an application which transforms the X–Y–Z patient specific coordinates of CSM stimulation sites into the common MNI (Montreal Neurological Institute) neuroanatomical coordinate system (Evans et al., 1993). Such spatial normalization is necessary in order to account for anatomical variability among individual brains. The transformation data are generated offline by a non-linear warping procedure in SPM that registers the structural MRI image volume of the patient to an MRI image volume of a population average brain. The warping procedure specifies, for each voxel in the MRI volume, the deformation necessary to transform it from patient-specific coordinates to normalized space. A transformation service (**Figure 1**) applies this deformation to transform a set of input patient-specific coordinates (typically stimulation sites retrieved from the CSM database) into the corresponding output MNI coordinates. The transformation server is wrapped in a web service that returns the result in XML format (**Figure 1**).

Foundational model of anatomy ontology. The Foundational Model of Anatomy (FMA) is a symbolic representation (ontology) of the taxonomic and structural relationships that comprise

canonical human anatomy (Rosse and Mejino, 2003). The FMA is useful, in DXBrain, for augmenting cross-source data alignment. This goes beyond direct equivalence mapping (i.e. cerebral cortex in source A is equivalent to cortex cerebri in source B) to identifying "related" anatomical entities (i.e. that the middle temporal gyrus is a part of the temporal lobe). Such relationships can be used to develop "intelligent" queries in DXBrain that, for example, find all CSM sites located anywhere in the temporal lobe, as we show in Section 'Consulting an Ontology to "Intelligently" Filter by Anatomical Location'. The FMA demonstrates the utility of using domain ontologies to facilitate semantic interoperability (whereas the transformation server in the previous section is a method for providing spatial interoperability through anatomical normalization).

As is the case with other sources the FMA must be made available via a web service interface that returns XML results in response to queries such as, "Find the parts of the temporal lobe". We have developed several such query interfaces (Mork et al., 2003; Stalder and Brinkley, 1999), the most recent being an ontology web service that accesses an OWL representation of the FMA, where OWL is a representation language for the semantic web (World Wide Web Consortium, 2005c).

The service accepts queries in the vSparQL query language (vSparQL Processor in **Figure 1**), an extension we are developing to the standard W3C recommended SparQL query language (World Wide Web Consortium, 2005b), that facilitates the creation of views over semantic web ontologies (Brinkley et al., 2006; Detwiler et al., 2008; Shaw et al., 2008). Since the result of a vSparQL view query is RDF (resource description framework) (World Wide Web Consortium, 2005a), which in turn is valid XML, the results of such a query can be included in a distributed XQuery. In fact the results of any standard SparQL query created with the SparQL "CONSTRUCT" statement can be included as a DXBrain source.

Distributed XQuery processor (DXQP)

DXQP (**Figure 1**) is responsible for processing DXQueries. It delegates source specific sub-queries to the appropriate source wrappers, and it handles local query operations such as cross-source joins, result filtering, and post processing. DXQP is described in its own project web page, and is available for download (Structural Informatics Group, 2007a). The primary component of DXQP is an XQuery library, which supports calls to our data source web service wrappers. Library functions are provided for working with both SOAP and simple HTTP *get* web services in order to allow inclusion of as many sources as possible.

SOAP services. SOAP (simple object access protocol) is a set of conventions for invoking code using XML over HTTP (W3C, 2008). SOAP provides a standardized way of calling remote program methods. However, XQuery does not have any built in means of communicating with SOAP services. To enable DXQP to communicate with SOAP source wrappers, we used an extension provided by the XQuery processor Saxon (Kay, 2008). Saxon is a free XQuery processor, written entirely in the Java programming language. Saxon facilitates the invocation of Java methods as external XQuery functions. We implemented a Java library, Anglo, to communicate with our SOAP source wrappers. We then added functions to the DXQP XQuery library that enable us to construct Anglo calls directly from an XQuery. While we endeavored to maintain compatibility with all standards-compliant XQuery processors, this extension works only on the Saxon processor (and possibly on others which offer similar support for Java calls, such as QEXO; GNU, 2008).

The primary user-callable XQuery function in this case is called dxq:xqueryWS, which takes as input the URL of the source web service, the port, the web service method name, and the query. This function then calls the appropriate Java-coded function in Anglo:

```
declare function dxq:xqueryWS(
$wsdlURL as xs:string,
$portName as xs:string,
$callName as xs:string,
$query as xs:string) as node()
{
anglo:xquery($wsdlURL,$portName,$callName,$query)
};
```

Get services. In addition to SOAP style services, DXQP also supports simple HTTP *get* web services where the query is encoded in the URL, and results are returned in XML. In the DXQP XQuery libraries we enabled interaction with simple *get* style source wrappers through the use of XQuery's built in doc($url) function. This function allows an XQuery to refer to any web accessible XML document. Because simple *get* services are accessible via a URL, and because they return XML results, they appear to the XQuery processor as XML documents. By passing queries to simple *get* source wrappers as URL arguments, the wrapper can dynamically generate XML results (a query shipping approach).

The primary user-callable XQuery function in this case is called dxq:xquery, which takes as input the server web address and the query. It constructs a URL by concatenating the server address, the web service method (called XQuerylet), and the hex-encoded query, and then calls the XQuery doc function to execute the query:

```
declare function dxq:xquery(
$server as xs:string,
$query as xs:string) as node()
{
let $queryString :=  encode-for-uri($query)
let$url := concat($server, "/XQuerylet?query =",
$queryString)
return doc($url)
};
```

Additional DXQP processing. In addition to delegating sub-queries to the appropriate source wrappers, DXQP can also process some portions of an XQuery locally. This enables DXQP to perform query operations such as cross-source data joins, result filtering, and other sorts of post processing tasks (such as in-query analysis).

Web server application

The DXBrain web server application (**Figure 1**) is an application server for integrating the multimodality brain mapping

data sources described in Section "Source Data and Source Wrappers". DXBrain communicates with DXQP [see "Distributed XQuery Processor (DXQP)"], which it currently incorporates as a Java library, enabling it to process DXQueries over the source network.

DXBrain is built as a dynamic web application using Java JSP and Servlet technologies. As such it presents a page-based navigational metaphor. A small number of pages allow users to perform the required operations: authentication, query construction, storing and retrieving queries, executing queries, and displaying query results. These pages communicate with the data sources using the DXQP libraries, as well as a local database for storing user authentication information and saved queries. Query results can also be piped to an external 3-D visualization application we developed called MindSeer (Moore et al., 2007).

Creating queries. Queries may be created in several ways. The simplest method allows users to compose their own custom DXQueries by simply entering them manually. This input method is flexible and powerful, but it is also requires users to first understand the XQuery language.

A second method provides a web form, which allows users to select from a set of available query parameters, as for example, codes describing the particular type of error during CSM stimulation (**Figure 2**). These options are inserted into pre-defined portions of a template query (a prewritten XQuery not crafted by the user). This method allows users to compose some queries without actually writing any XQuery, but it supports only a very small subset of possible queries. This input method serves mostly as a demonstration of the usability gained by layering an easy to use, use-case specific user interface on top of an expressive and flexible query engine.

Storing and retrieving queries. We have found that the most useful means for composing queries is to retrieve a previously stored query and use it as a template for creating a new one. Any query, whether it is generated anew, created from a template via the selection interface, or retrieved from existing queries, may be saved in a local query database. Each saved query is given a title and description so that it may be retrieved from a list of stored queries, organized by the user. For example, **Figure 3** shows a portion of one such stored query, called "csm_view_code_fma_fmri". Any of the Title, Description or Query fields may be edited by the user who owns the query. The query may be marked public, allowing other users to access it in the list of stored queries, or it can be made private (which is usually done while the query is being developed).

Executing queries. No matter what the input method a query is executed by clicking on one of the output format buttons at the bottom of the page (**Figure 3**). In each case the query is sent to DXQP for local execution or dispatch of query fragments to the separate sources, after which the consolidated XML results returned by DXQP are transformed if necessary into the requested output format.

RESULTS

We illustrate DXBrain by showing how it is used to process the following query:

> "Find all female patients who made a semantic naming error during cortical stimulation mapping (CSM). For each of these patients find all CSM stimulation sites where at least one such error was made and that are located in the temporal lobe of the brain. For each of these sites return the normalized 3-D coordinates so that they may be compared. In addition, for each patient with at least one such error, return the associated fMRI study that was done using the same object identification protocol, if there is one."

Simple modifications of this query would, for example, substitute male for female, and syntactic error for semantic error. The purpose of this query is to determine if different types of language processing are localized in different parts of the brain for different population subgroups (males versus females in this case) and whether there are any correlations between different measures of language such as CSM and fMRI.

This query needs to integrate CSM and fMRI data stored in the CSM and fMRI data sources. It also needs to access the transformation service to normalize patient coordinates. In addition, the query asks for stimulation sites (stimsites) that are located in the temporal lobe. However, in the CSM database stimsites are annotated with anatomical names that are of a finer granularity than "temporal lobe" (i.e. "Middle part of superior temporal gyrus"). Thus, the query also needs to query the FMA to determine the parts of the temporal lobe, and then only retain stimsites that are annotated by any part of the temporal lobe.

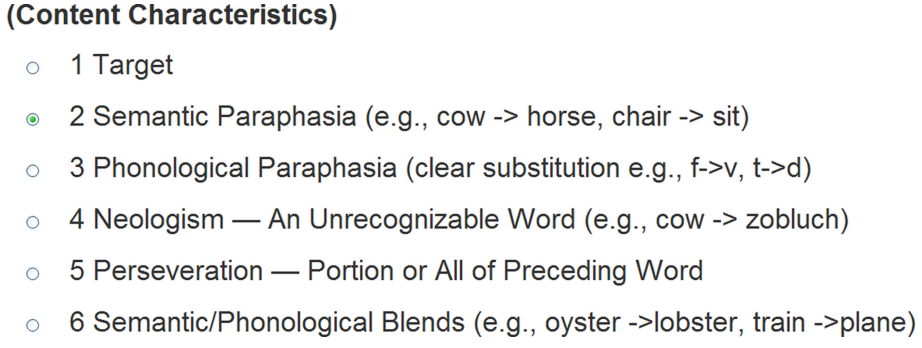

FIGURE 2 | Portion of DXBrain template query GUI interface.

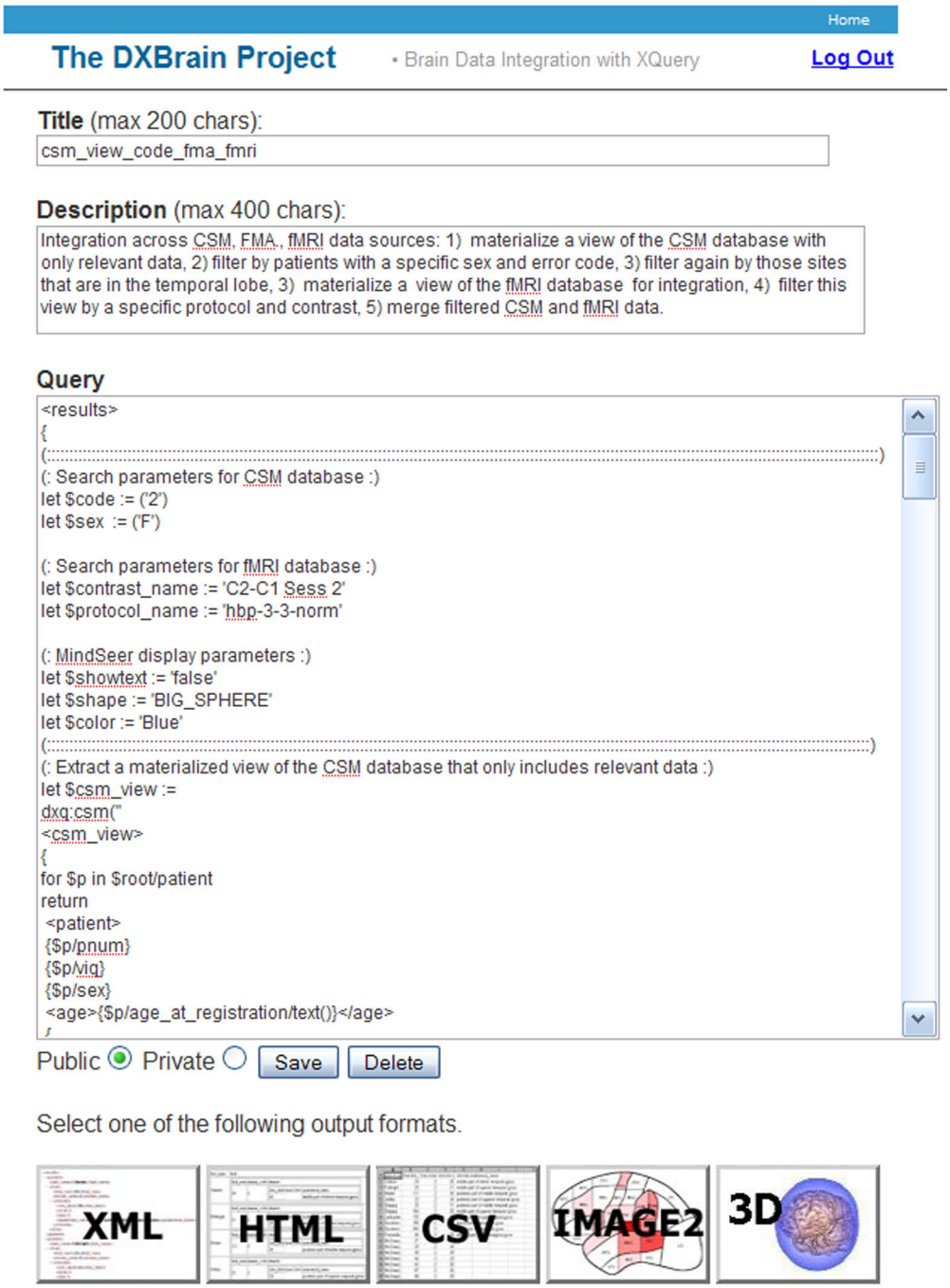

FIGURE 3 | Saved query.

EXAMPLE DXQUERY

The Query box of **Figure 3** shows a snippet of a DXQuery that answers the above query. The full DXQuery, which is 251 lines long, is available as a demo that can be examined and run (Structural Informatics Group, 2008). Note, however, that the results returned by the demo will be minimal since full access to the data is restricted to authorized users. The next few sections explain snippets of this query.

Parameter assignments

As shown in the portion of the query displayed in **Figure 3** saved queries often start with a set of parameter assignments, which represent search or display parameters. In the sample query, $code is a list of CSM error codes for searching the CSM database (as in **Figure 2**), $contrast_name and $protocol are parameters for searching the fMRI database, and $showtext, $color and $shape are parameters for our 3D visualization tool,

Mindseer. In future work these parameters could be set in an automatically-generated parameter selection interface like that shown in **Figure 2**.

A view of the CSM database

The next part of the query snippet shown in **Figure 3** sets a local variable $csm_view to be an in-memory XML view of the CSM database that includes only those elements desired for the subsequent searches. This view may also involve renaming of some of the elements, as for example, <age> for <age_at_registration>. The constructed view query is executed by means of the dxq:csm XQuery function, which is a shortcut for the dxq:xquery HTTP *get* style function described in Section "Get Services". This shortcut allows users to call the CSM service without having to remember its URL. In DXBrain several of the other web services (fMRI, transformation) are similarly encapsulated.

Filtering the CSM view by sex and error code

The following sections include snippets of the full query that are not visible in **Figure 3**.

The CSM view is next filtered by the selected $sex parameter and the particular set of error codes specified by the $code parameter, the result of which is assigned to the $sex_code_filtered variable:

```
let $sex_code_filtered: =
  <sex_code_filtered>
  {for $p in $csm_view/csm_view/patient
  where $p/sex = $sex
    and $p/surgery/stimsite/trial/trialcode/
    term/abbrev/text() = $code
  return
    <patient>
      ...
```

The result of this subquery is that $sex_code_filtered now points to an in-memory XML fragment containing only females who made a stimulation error of type 2 anywhere in the brain.

Consulting an ontology to "intelligently" filter by anatomical location

To limit the location to just the temporal lobe the separate FMA source ontology is first consulted to determine the transitive closure (all parts of parts) of the Temporal lobe. In this case the language of the source-specific sub-query is not XQuery, but rather vSparQL (see "Foundational Model of Anatomy Ontology"). The following simplified snippet shows a small part of the OWL representation of the FMA ontology (which has over 72,000 concepts and 2 million relationships).

```
fma:Temporal_lobe fma:regional_part
fma:Middle_temporal_gyrus.
fma:Middle_temporal_gyrus fma:regional_part
fma:Posterior_part_of_middle_temporal_gyrus.
fma:Posterior_part_of_middle_temporal_gyrus
fma:Preferred_name fma:fma_term_08300.
fma:fma_term_08300 fma:name "Posterior part of
middle temporal gyrus".
```

The corresponding vSparQL query, which is assigned to the variable $query in the distributed XQuery, is shown below.

```
(: Define the parameters of the ontology web
service :)
let $wsdlURL :=  "http://.../VSparQLService.wsdl"
let $serviceName := "VSparQLService"
let $methodName :=  "executeQuery"
(: Define the subquery in extended SparQL :)
let $query :=
'PREFIX fma: <http://.../fma_2_0#>
PREFIX gleen:<java:edu.washington.sig.gleen.>
PREFIX qv:<http://sig.biostr.washington.
edu/query_view#>
CONSTRUCT { fma:Temporal_lobe qv:hasPartName
?part_name. }
FROM <http://.../fma_2_0>
WHERE
{
  fma:Temporal_lobe gleen:OnPath
    ("([fma:regional_part]|[fma:constitutional_
    part]) +" ?part).
  ?part gleen:OnPath ("[fma:Preferred_name]/
  [fma:name]" ?part_name).
}'
(: Execute the query and return only temporal
lobe parts from the XML result tree :)
let $temporalLobeParts :=  dxq:xqueryWS($wsdlURL,
$serviceName, $methodName, $query)//
qv:hasPartName
```

As noted in Section "Foundational Model of Anatomy Ontology" vSparQL (Shaw et al., 2008) is a set of extensions to SparQL we are developing to enable the creation of views over large ontologies like the FMA. In particular this query uses the Gleen regular expression library (Detwiler et al., 2008) to follow arbitrary paths in ontologies. For example, in the above query the search starts with the node called "Temporal_lobe", follows any number of regional_part or constitutional_part links (which together define the generic parts), and for each of these, finds the string representing the preferred name of that part. This string is then assigned to each element <hasPartName> of the result. Since the annotations of the CSM database were controlled to use these exact strings then the result of this query may be used directly to match against the CSM data. In a more complex situation a separate mapper web service associated with a particular data source could be used to convert widely accepted ontology terms to local terms. The DXBrain architecture would treat such a mapper as just another data source.

The list of parts of the Temporal lobe is then used to further filter the $sex_code_filtered result by selecting only those patients who have stimsites that are annotated with at least one of the parts of the temporal lobe (via the anatomical_name child element of stimsite). The result of this filtering operation is assigned to variable $fma_filtered. Its effect is to restrict the retrieved stimsites to only those that are in the temporal lobe, without requiring the user to know the specific parts of the

temporal lobe that were used to annotate the data in the original CSM database:

```
let $fma_filtered :=
  <fma_filtered>
  {for $p in $sex_code_filtered/patient
   where $p/surgery/stimsite/anatomical_name =
   $temporalLobeParts
  return
   <patient>
…
```

Querying a second data source

Next, a view of the fmri database is assigned to the variable $fmri_view via dxq:xbatch, a shortcut function that directs xqueries to the fMRI Wix wrapper in **Figure 1**. As in csm_view, many of the elements in the original database are deleted or renamed, as for example, <Subject> becomes <patient> to conform to the usage in the CSM database.

```
let $fmri_view :=  dxq:xbatch("
  <fmri>
   {for $subject in $root//Subject
   let $subjectID :=  data($subject/
   @Subject_ID)
   let $pnum :=  replace($subjectID,'P',")
   order by $pnum
   return
   <patient>
   …
```

This view is then filtered and assigned to variable $fmri_filtered so that only fMRI protocols and contrasts as specified by the search parameters are retained:

```
let $fmri_filtered :=
  <fmri>
   {for $patient in $fmri_view//patient
   where $patient/protocol/name/
   text() = $protocol_name and
   $patient/protocol/contrast/name/
   text() = $contrast_name
   return
   <patient>
   …
```

Merging results from the two data sources

The $fmri_filtered result is then joined with the $fma_filtered result, with the result that available fMRI contrasts are included with the results from the CSM database.

```
let $merged :=
  <merged>
  {for $p in $fma_filtered//patient
   return
   <patient>
     …
    {for $s in $fmri_filtered//patient
    where $p/pnum = $s/pnum
     return
      $s/contrast
    }
  </patient>
  }
</merged>
```

Finally, the merged result, along with a count of patients, is returned as the overall result of the query:

```
return
  <patients>
   <count>{count($merged/patient)}</count>
  {$merged/patient}
</patients>
```

DISPLAYING QUERY RESULTS

DXBrain provides multiple formats for displaying the results of a dxquery. The textual output formats XML, HTML and CSV are generic to any data source. The Image2 and 3D formats, which layer results on a 2D and 3D representation of the brain respectively, are specific to the DXBrain application. The buttons shown at the bottom of **Figure 3** are used to execute the query in the Query box, and to transform the results to the requested output format.

Displaying results in XML

XML is the native format of query results from DXQP. If the user selects the XML output format, he or she is presented with the raw results, as they are received from DXQP, without any additional post-processing. For example, **Figure 4** shows a snippet of the XML results obtained from running the sample query. In this case 16 females made semantic paraphasia errors during CSM that were located in the temporal lobe. The figure shows one of these, P175, together with the one stimsite where the error was made, the magnet (patient-specific coordinates) of that site, the assigned anatomical name, "Middle part of middle temporal gyrus", which is part of the temporal lobe, and the associated CSM trials during which the error was made. For example, in trial 69 the patient was presented with a picture of a chicken while site 28 was stimulated. The response, "hen, rooster", was coded as a semantic error (trialcode 2).

In addition, the full local pathname to an associated fMRI contrast file, spmT_0012.hdr, is also shown. This file represents an fMRI study with similar object identification task as that used in the CSM study, the goal being to compare the CSM and fMRI results.

Displaying results in HTML and CSV

The HTML and CSV output options allow users to view DXQP results transformed from XML into HTML or CSV, where CSV is a useful input format for spreadsheet programs like Excel. In both cases we generated the transformed output by means of internal XQueries that are applicable to *any* well-formed XML file, regardless of schema.

Figure 5 shows a snippet of the HTML generated from the XML file shown in **Figure 4**, whereas **Figure 6** is a snippet of the CSV output from the XML file, after being loaded into Excel.

```
- <patient>
    <pnum>175</pnum>
    <viq/>
    <sex>F</sex>
    <age>31</age>
  - <surgery>
    - <stimsite>
        <ShowText>false</ShowText>
        <Color>Blue</Color>
        <Shape>BIG_SPHERE</Shape>
        <type>Surgical</type>
        <site_label>28</site_label>
      - <magnet_coordinates>
          <ant_coord>10.0739</ant_coord>
          <sup_coord>8.84224</sup_coord>
          <right_coord>-63.0056</right_coord>
        </magnet_coordinates>
        <anatomical_name>Middle part of middle temporal gyrus</anatomical_name>
      - <trial>
          <trial_num>69</trial_num>
          <stimulated>Y</stimulated>
          <item>chicken</item>
          <patient_response>"hen, rooster"</patient_response>
          <trialcode>2</trialcode>
        </trial>
      </stimsite>
    </surgery>
  - <contrast>
      <name>C2-C1 Sess 2</name>
    - <File>
        file:/usr/local/data/data11/hbp/P175/spm/stats/S101/results_P175_sess1_hbp-3-3-norm/spmT_0012.hdr
      </File>
    </contrast>
  </patient>
```


FIGURE 4 | XML output.

pnum	viq	sex	age	surgery
text		text	text	stimsite
175		F	31	

ShowText	Color	Shape	type	site_label	magnet_coordinates			anatomical_name	trial	
text	text	text	text	text	ant_coord	sup_coord	right_coord	text	trial_num	stimulated
false	Blue	BIG_SPHERE	Surgical	28	text	text	text	Middle part of middle temporal gyrus	text	text
					10.0739	8.84224	-63.0056		69	Y

FIGURE 5 | HTML output.

Displaying results as 2-D images

In addition to these general result formats DXBrain provides two additional means for visualizing query results. These two output methods are specific to the DXBrain data network. The first, a 2D visualization (Image2 button in **Figure 3**), shows a segmented brain schematic, where each region in the diagram is colored according to the number of retrieved stimulation sites that are annotated with the name of that region. The region labels are mapped to terms from the FMA ontology. For example MSTG is an abbreviation for Middle part of superior temporal gyrus. **Figure 7A** shows

patient.age	patient.pnum	patient.sex	stimsite.anatomical_name	magnet_c	magnet_c	magnet_c	stimsite.site_label	trial.item	trial.patient_response	trial.trial_num	trial.trialcode
38	40	F	Middle part of superior temporal gyrus	15.2846	-61.117	43.2242	40	pig	um...(fillers)-cow!	237	B
38	40	F	Middle part of superior temporal gyrus	15.2846	-61.117	43.2242	40	pig	um...(fillers)-cow!	237	2
46	52	F	Anterior part of superior temporal gyrus	24.9601	-63.551	-5.8579	23	dress	dressers	106	2
46	52	F	Anterior part of superior temporal gyrus	24.9601	-63.551	-5.8579	23	dress	dressers	106	FSE
46	52	F	Anterior part of superior temporal gyrus	24.9601	-63.551	-5.8579	23	table	t.v.	161	A
46	52	F	Anterior part of superior temporal gyrus	24.9601	-63.551	-5.8579	23	table	t.v.	161	2
46	52	F	Middle part of superior temporal gyrus				30	cow	pig	66	B
46	52	F	Middle part of superior temporal gyrus				30	cow	pig	66	2
46	52	F	Middle part of inferior temporal gyrus	-6.1483	-70.509	-16.367	26	lion	tiger	24	2
45	64	F	Polar part of inferior temporal gyrus	9.94473	-52.241	-10.947	20	tiger	lion...uh,tiger	2	2
45	64	F	Polar part of inferior temporal gyrus	9.94473	-52.241	-10.947	20	tiger	lion...uh,tiger	2	S
49	84	F	Posterior part of middle temporal gyrus				26	dresser	drawer	111	2
49	84	F	Posterior part of middle temporal gyrus				26	dresser	drawer	111	FSE

FIGURE 6 | CSV output displayed in Excel.

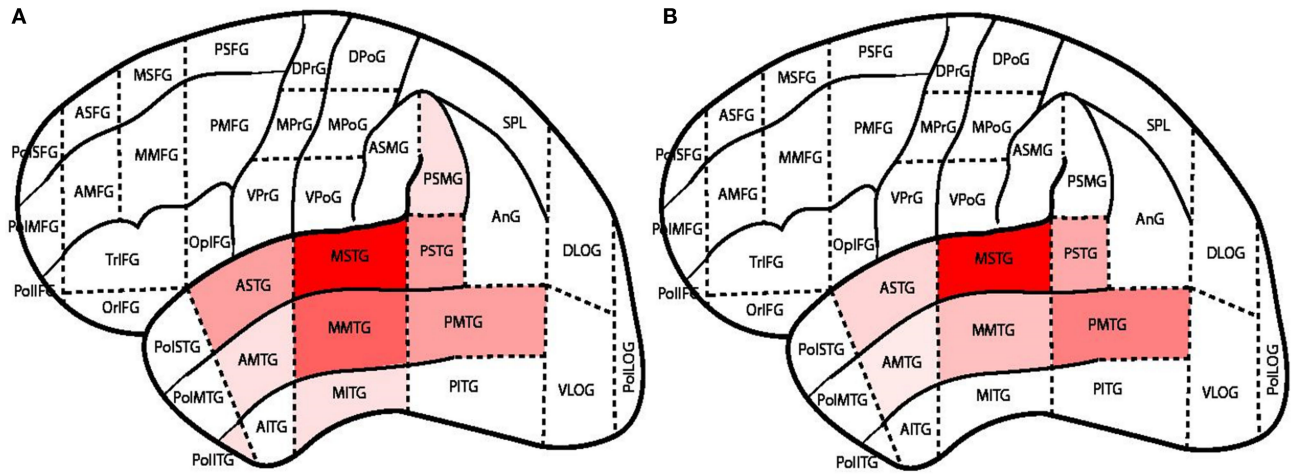

FIGURE 7 | 2-D image output. Intensity of colored regions is proportional to the number of sites in that region returned by the query. **(A)** Females, **(B)** Males.

such a visualization for female patients, and **Figure 7B** shows male patients after re-running the example query with the $sex parameter changed to "M".

Displaying results as 3-D visualizations

The second visualization method pipelines results from DXBrain to an external 3D visualization suite we developed, called MindSeer (Moore et al., 2007). It does this by first connecting to the Transformation service (**Figure 1**) to convert the patient-specific (magnet) coordinates of each stimsite, as returned in **Figure 4**, to normalized coordinates. It then invokes MindSeer via Java Web Start, with parameters that include the results returned by running the query.

MindSeer is both a standalone and a client-server Java3D program that supports the visualization of spatial data, including spatial data resulting from DXBrain queries. For example, stimulation sites from multiple patients can be viewed together spatially normalized, on a common brain. Additional data modalities, such as fMRI, can be overlaid within the same visualization. Two canonical brain atlases are available within which MindSeer can display DXBrain results, the Collin brain and the Average MNI Brain (Evans et al., 1993). Results can also be viewed on any of the individual patient's brains as well. Such a capability could be useful for surgical planning, in order to show likely regions of language, based on population studies, which could be further explored at the time of surgery.

Figure 8A shows the normalized stimulation sites retrieved as a result of the sample query as large blue spheres (one of the parameters of the query, **Figure 3**). In addition, the single retrieved fMRI study (based on the query parameters $contrast and $protocol in **Figure 3**) is shown. In this case the intensity values of the fMRI volume (which represent areas of the brain that are activated as a result of the task) are used to color the nearest surface patches on the canonical brain surface, thereby allowing comparison with the CSM sites. In this case the cold colors (like blue) in the color scheme mean the area had decreased activation relative to a control task (fixation), and warm colors (like orange) means the area had increased activation. It should be noted that no conclusions as to the relationship between fMRI and CSM measures of language can be inferred from this one example, and in fact, other studies suggest that there may in fact not be a relationship. The point for this paper is that the tools we have developed facilitate such comparisons.

Figure 8B shows the results of running the sample query in which males are substituted for females, as in **Figure 7B**. In this case no fMRI study is available that matches the search criteria (the fMRI database is currently only sparsely populated).

Although the CSM data shown in **Figures 7 and 8** suggest that semantic processing is more posterior in the temporal lobe for males than for females, these suggestions need to be validated both by statistical analysis and by looking at other sources of data. The CSV output format (**Figure 6**) facilitates this sort of analysis by external statistical packages, and the ability of DXBrain, the FMA and MindSeer to query and integrate multiple sources of data should allow these other sources to be more readily accessed than would be the case if they each had to be accessed manually.

ADDING A NEW SOURCE

One of the primary advantages of the distributed query approach is that it is very easy to add new data sources to the network, even those that were not anticipated during the original system deployment. As DXBrain does not enforce a centralized schema, and because it places few requirements or restrictions on new data sources, adding such sources is relatively simple. To make it even easier we provide the WIX tool (see "Source Data and Source Wrappers") to assist users in the task of source deployment.

Using the WIX tool, we performed a small test to determine the difficulty of adding a new, unanticipated source to the network. For the source, we used an XML export of the EndNote library for this paper. While this source adds little value to the network, it was chosen due to its dissimilarity with existing sources. We also downloaded the DTD file containing the general EndNote XML export schema. We launched WIX, selected our XML and Schema files and then clicked the "Generate War" button, which produces a .war file (**W**eb **AR**chive) suitable for deployment in any J2EE web application container (such as Apache Tomcat). This process took roughly 1 min. We used Tomcat's manager application to deploy our newly generated .war file, which again took about a minute. Our source is now available for querying from DXBrain.

To test our new source deployment, we went to the DXBrain "New Query" page and issued a query against our new source. The query we chose finds the titles of all references where the author name contains the word "Bales". Here is our test query, which took about 5 min to generate as we needed to refer to the schema when creating the query, and our results:

```
let $endnote_query := "
<results>
{
   for $record in $root//record
   for $record_style in $record//author/style
   where contains($record_style,'Bales')
   return
   <title>{$record//title/style/text()}</title>
}
</results>"
return
   dxq:xquery
   ("http://xiphoid:8080/DXBrain_EndNote/",
$endnote_query )
-------------------------------------------------

<?xml version = "1.0" encoding = "UTF-8"?"
<results>
   <title>A framework for XML-based integration
   of data, visualization and analysis in a
   biomedical domain</title>
</results>
```

DISCUSSION

In this paper we have described a distributed XML-based approach to data integration, and have shown its implementation within a specific application in brain mapping. The approach is lightweight and efficient, in that it is very easy to add a new data source, and query processing is relatively fast. However a cognitive burden is moved from the administrator to the user, who must now understand all the source schemata. We have partially reduced this burden by allowing the use of saved queries, which can be re-executed at

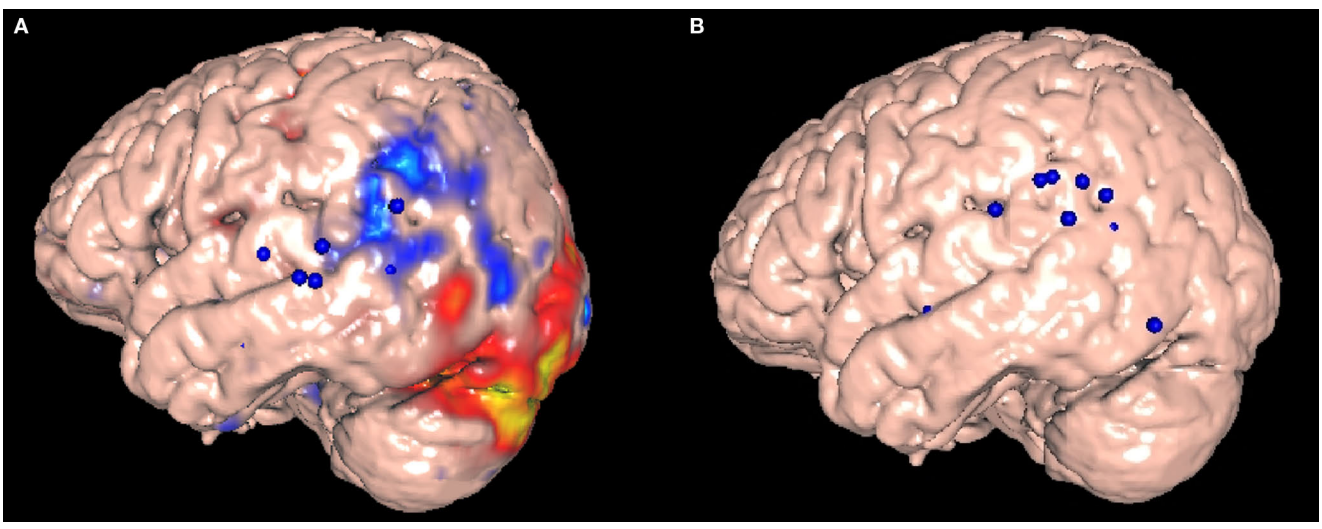

FIGURE 8 | 3-D Visualization of retrieved CSM sites mapped onto a canonical brain atlas. (A) Females, also showing a related fMRI study, **(B)** Males.

any time with new parameters that may or may not be generated by custom user interfaces. Thus, we envision that one or more XQuery experts, who know the data sources, will be required to generate the saved queries, whereas end-users will be able to simply re-run the queries or use simple graphical interfaces to complete pre-built template queries. In practice we have found that if the query results are useful enough, and if there are enough example saved queries to serve as templates, the collaborating biologists do indeed take the time to learn enough XQuery to obtain results. In fact some of the saved queries on our demo website were generated by one such collaborating biologist. In contrast, it is unlikely that a collaborating biologist would be willing to take the considerable time needed to create and maintain a data warehouse or mediated system if all they want is to get a new result for a single publication.

In spite of the limitations we have found the system to be very useful in its present state as a way to rapidly query disparate data sources. In fact many of the sources we have queried in this way are not even related to brain mapping, as, for example the EndNote library described in Section "Adding a New Source". Thus, in our future work we will generalize our approach, removing the dependencies on brain mapping and relegating these to a plugin mechanism. In addition we will develop methods for adding additional sources, without requiring the developer to know the schema of every source, through the creation of views over multiple sources, which can themselves be queried as though they were a single source. Such an approach is analogous to views in relational databases, and similar to our current work in view generation over ontologies (Brinkley et al., 2006).

As the number of sources increases methods for semantic interoperability will become increasingly important, so the semantic web query methods described in Section 'Consulting an Ontology to "Intelligently" Filter by Anatomical Location' will need to be enhanced. In addition, since our eventual goal is to allow the end user to create queries without the need to understand the complex XQuery language, we will need to develop graphical XQuery generation languages similar to a prototype we have developed but not yet integrated in DXBrain (Li et al., 2007). Finally, as the number of sources becomes larger, or as the data become ready for submission to large central repositories or federations such as BIRN, it will be necessary to develop interfaces such that it is easy to either import the results of a distributed query into one of these systems, or to treat the distributed query engine as a processing node in the larger grid.

Many of the components of DXBrain are analogous to those in other data integration systems: the use of wrappers over data sources, a distributed query processor, semantic web and other technologies for interoperability, and user interfaces. An important advantage of XQuery and its distributed extensions is that it is a complete programming language, thus allowing many of these components to be developed within a common framework, and then saved as modules (in the form of saved queries). Thus, as methods for re-using these modules and for graphically generating them become available it should be possible to build from the bottom-up networks of data sources from small groups of collaborating labs. These networks could in turn be combined together through additional saved query modules, and eventually incorporated as nodes in the larger national grids. Although additional research is needed before this kind of bottom-up creation of an information sharing network becomes available the current version of DXBrain, especially the downloadable modules, should be of use now by others who wish to create local shared networks of data. Tools to create such networks should help to fill the gap between the mostly *ad hoc* methods currently used by small-scale investigators and the large-scale heavyweight methods employed by projects such as BIRN.

ACKNOWLEDGEMENTS

This work was funded by NIH grants DC02310 and HL087706. In addition to the authors, several other individuals have been involved in various aspects of this work, including Chris Re, Nathan Bales, Stacy Tang, Hao Li, Erin Gibson, Brandon Loudermilk, and Ettore Lettich. We thank Natasha Noy, at the Stanford National Center for Bioontology, for converting the Foundational Model of Anatomy to OWL.

REFERENCES

Apache Software Foundation. (2007). Apache Tomcat. 2007. Available at: http://tomcat.apache.org/.

Bales, N., Brinkley, J, Lee, E. S., Mathur, S., Re, C., Suciu, D. (2005). A Framework for XML-Based Integration of Data, Visualization and Analysis in a Biomedical Domain. Trondheim, Proceedings, Third International XML Database Symposium (XSym 2005), pp. 207–221.

Brinkley, J. F. (2008). University of Washington Integrated Brain Project, Home Page. Available at: http://sig.biostr.washington.edu/projects/brain.

Brinkley, J. F., Jakobovits, R. M., Poliakov, A. V., Martin, R. F., Gibson, E. R., Corina, D. M., Ojemann, G. A. (2004). An experiment management system for cortical stimulation mapping data. San Diego, Society for Neuroscience Annual Meeting, p. 1032.12.

Brinkley, J. F., Myers, L. M., Prothero, J. S., Heil, G. H., Tsuruda, J. S., Maravilla, K. R., Ojemann, G. A., Rosse, C. (1997). A structural information framework for brain mapping. In: Neuroinformatics: An Overview of the Human Brain Project, S. H. Koslow and M. F. Huerta, eds (Mahwah, Lawrence Erlbaum), pp. 309–334.

Brinkley, J. F., Suciu, D., Detwiler, L. T., Gennari, J. H., Rosse, C. (2006). A framework for using reference ontologies as a foundation for the semantic web. *Proc. AMIA Annu. Fall Symp.* 2006, 96–100.

Detwiler, L. T., Suciu, D., Brinkley, J. F. (2008). Regular paths in SparQL: querying the NCI thesaurus. *Proc. AMIA Annu. Fall Symp.* 2008, 161–165.

Evans, A. C., Collins, D. L., Mills, S. R., Brown, E. D., Kelly, R. L., Peters, T. M. (1993). 3D Statistical neuroanatomical models from 305 MRI volumes. *Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf.* 1993, 1813–1817.

Fernandez, M., Jim, T., Morton, K., Onose, N., Simeon, J. (2007). DXQ: A Distributed XQuery Scripting Language. Beijing, XIME-P-2007: 4th International Workshop on XQuery Implementation, Experience and Perspectives.

Fernandez, M., Kadiyska, Y., Morishima, A., Suciu, D., Tan, W. (2002). Silkroute: a framework for publishing relational data in XML. *ACM Trans. Database Technol.* 27, 1–6.

Gazzaniga, M. S. (2008). The fMRI Data Center. 2001. Available at: http://www.fmridc.org/.

GNU. (2008). QEXO – The GNU Kawa Implementation of XQuery. Available at: http://www.gnu.org/software/qexo/.

Hinshaw, K. P., Poliakov, A. V., Martin, R. F., Moore, E. B., Shapiro, L. G., Brinkley, J. F. (2002). Shape-based cortical surface segmentation for visualization brain mapping. *Neuroimage* 16, 295–316.

Kay, M. (2008). SAXON: The XSLT and XQuery Processor. Available at: http://saxon.sourceforge.net/.

Keator, D. B., Grethe, J. S., Marcus, D., Ozyurt, B., Gadde, S., Murphy, S., Pieper, S., Greve, D., Notestine, R., Bockholt, H. J., Papadopoulos, P. (2008). A national human neuroimaging collaboratory enabled by the

Biomedical Informatics Research Network (BIRN). *IEEE Trans. Inf. Technol. Biomed*. 12, 162–172.

Koslow, S., and Hyman, S. (2000). Human brain project: a program for the new millenium. *Einstein Q. J. Biol. Med.* 17, 7–15.

Laboratory of Neuro Imaging. (2008). LONI Image Database. Available at: http://www.loni.ucla.edu/Research/Databases/.

Li, X., Gennari, J. H., Brinkley, J. (2007). XGI: a graphical interface for XQuery creation. *Proc. AMIA Annu. Fall Symp.* 2007, 453–457.

Martone, M. E., Zhang, S., Gupta, A., Qian, X., He, H., Price, D. L., Wong, M., Santini, S., Ellisman, M. H. (2003). The cell-centered database: a database for multiscale structural and protein localization data from light and electron microscopy. *Neuroinformatics* 1, 379–395.

Miller, P. L., Nadkarni, P., Singer, M., Marenco, L., Hines, M., Shepard, G. (2001). Integration of multidisciplinary sensory data: a pilot model of the Human Brain Project approach. *J. Am. Med. Assoc.* 8, 34–48.

Moore, E. B., Poliakov, A. Lincoln, P., Brinkley, J. (2007). MindSeer: a portable and extensible tool for visualization of structural and functional neuroimaging data. *BMC Bioinformatics* 8, 389.

Mork, P., Brinkley, J. F. Rosse, C. (2003). OQAFMA Querying Agent for the Foundational Model of Anatomy: a prototype for providing flexible and efficient access to large semantic networks. *J. Biomed. Inform.* 36, 501–517.

Nature Neuroscience. (2000). A debate over fMRI data sharing. *Nat. Neurosci.* 3, 845–846.

Ojemann, G., Ojemann, J., Lettich, E., Berger, M. (1989). Cortical language localization in left, dominant hemisphere: an electrical stimulation mapping investigation in 117 patients. *J. Neurosurg.* 71, 316–326.

Oster, S., Langella, S., Hastings, S., Ervin, D., Madduri, R., Phillips, J., Kurc, T., Siebenlist, F., Covitz, P., Shanbhag, K., Foster, I., Saltz, J. (2008). caGrid 1.0: an enterprise Grid infrastructure for biomedical research. *J. Am. Med. Inform. Assoc.* 15, 138–149.

Ozsu, T., and Valduriez, P. (1999). Principles of Distributed Database Systems. Prentice Hall.

Poliakov, A., Hertzenberrg, X., Moore, E. B., Corina, D., Ojemann, G. A., Brinkley, J. F. (2007). Unobtrusive integration of data management with fMRI analysis. *Neuroinformatics* 5, 3–10.

Re, C. (2006). SilkRoute II – Efficient relational publishing to XML. 2006. Available at: http://silkroute.cs.washington.edu/.

Re, C., Brinkley, J., Hinshaw, K., Suciu, D. (2004). Distributed XQuery. Proceedings of the Workshop on Information Integration on the Web (IIWeb), pp. 116–121.

Rosse, C., and Mejino, J. L. V. (2003). A reference ontology for bioinformatics: the Foundational Model of Anatomy. *J. Bioinform.* 36, 478–500.

Shaw, M., Detwiler, L. T., Brinkley, J. F, Suciu, D. (2008). Generating application ontologies from reference ontologies. *Proc. AMIA Annu. Fall Symp.* 2008, 672–676.

Society for Neuroscience. (2008). Neuroscience Database Gateway. 2008. Avaialable at: http://ndg.sfn.org/.

Stalder, D. S., and Brinkley, J. F. (1999). The Digital Anatomist Foundational Model Server. Monterey, Perl Conference 3.0.

Structural Informatics Group. (2007a). DXQP – Distributed XQuery Processor. 2007. Available at: http://sig.biostr.washington.edu/projects/dxqp/.

Structural Informatics Group. (2007b). WIX: Web Interface for XQuery. 2007. Available at: http://sig.biostr.washington.edu/projects/wix/index.html.

Structural Informatics Group. (2008). DXBrain Demo Query. 2007 Available at: http://sig.biostr.washington.edu/projects/dxbrain/demoquery.html.

Van Essen, D. (2008). SumsDB. Available at: http://sumsdb.wustl.edu:8081/sums/dispatch.do?forward = index.

W3C. (2008). Simple Object Access Protocol (SOAP). Available at: http://www.w3.org/TR/soap/.

Wellcome Department of Cognitive Neurology. (2001). Statistical Parametric Mapping. Available at: http://www.fil.ion.ucl.ac.uk/spm/.

World Wide Web Consortium. (2005a). Resource Description Framework (RDF). 2005. Available at: http://www.w3.org/RDF/.

World Wide Web Consortium. (2005b). SparQL query language for RDF. 2005. Available at: http://www.w3.org/TR/rdf-sparql-query/.

World Wide Web Consortium. (2005c). The OWL Web Ontology Language. Available at: http://www.w3.org/TR/owl-features/.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 30 October 2008; paper pending published: 10 December 2008; accepted: 10 January 2009; published online: 30 January 2009

*Citation: Detwiler LT, Suciu D, Franklin JD, Moore EB, Poliakov AV, Lee ES, Corina DP, Ojemann GA and Brinkley JF (2009) Distributed XQuery-based integration and visualization of multimodality brain mapping data. Front. Neuroinform. (2009) **3**:2. doi: 10.3389/neuro.11.002.2009*