

Integrating Data-Mining Support into a Brain-Image Database Using Open-Source Components

Herskovits EH*, Chen R

Department of Radiology, University of Pennsylvania, Philadelphia, U.S.

*** CORRESPONDING AUTHOR:**

Department of Radiology
University of Pennsylvania
3400 Spruce Street
Philadelphia, PA 19104
United States
Telephone: (215) 215-662-6865
ehh@ieee.org (Edward Herskovits)

Received 13.02.2008
Accepted 20.03.2008
Advances in Medical Sciences
Vol. 53(2) · 2008 · pp 172-181
DOI: 10.2478/v10039-008-0009-9
© Medical University of Białystok, Poland

ABSTRACT

Purpose: Previously, we described our implementation of a brain-image database (BRAID), based on the proprietary object-relational database-management system (ORDBMS) Illustra [1]. In conjunction with our collaborators, we have used this database to manage and analyze image and clinical data from what we call image-based clinical trials (IBCTs). Herein we describe the results of redesigning BRAID using open-source components, and integrating support for mining image and clinical data from BRAID's user interface.

Material and Methods: We re-designed and re-implemented BRAID using open-source components, including PostgreSQL, gcc, and PHP. We integrated data-mining algorithms into BRAID, based on PL/R, a PostgreSQL package to support efficient communication between R and PostgreSQL.

Results: We present a sample clinical study to demonstrate how clinicians can perform queries for visualization, statistical analysis, and data mining, using a web-based interface.

Conclusion: We have developed a database system with data-mining capabilities for managing, querying, analyzing and visualizing brain-MR images. We implemented this system using open-source components, with the express goal of wide dissemination throughout the neuroimaging research community.

Key words: brain-image, database, data mining

INTRODUCTION

One of the principal goals of the brain-image database (BRAID) project is the development of the computational infrastructure necessary to support the management and analysis of the image-based clinical trials (IBCTs), in which medical imagery constitutes a critical component of the data collected [1]. In particular, BRAID is often used to support lesion-deficit analysis [2], in which the primary goal is to determine associations among locations of abnormal voxels, and outcome measures or predisposing factors. For a particular study, lesion-deficit data typically consist of 3-dimensional maps of subjects' lesions, demographics, predisposing factors, and outcomes.

We must perform two steps before we can incorporate images into BRAID: we must identify, or segment, brain lesions, and we must register the image data to a common spatial standard. Either human experts, such as neurologists or

neuroradiologists, or computer programs, may be employed to identify lesions. We then register the image data (including lesions) to a common spatial standard, such as an atlas. We employ image-warping methods, in which each voxel is individually displaced (e.g., [3]), for this purpose.

Once we have standardized image and clinical data, we integrate them into our database-management system (DBMS), to facilitate visualization and analysis of these data. Given the complexity and volume of data obtained from a typical IBCT, an image database is critical for management of these data. There are three general DBMS architectures that could be used to implement an image database: relational (RDBMS), object-relational (ORDBMS), and object-oriented (OODBMS) [4]. In their review of suitable DBMS architectures for neuroimaging, Diallo et al. reported that the OODBMS approach has disadvantages relative to an ORDBMS, such as the former's incompatibility with legacy DBMS data,

and absence of a standard [5]. Stonebraker's four-quadrant DBMS classification [6] supports this conclusion, suggesting that an ORDBMS may be the best choice when managing complex data and processing complex queries. We therefore chose to implement BRAID using an ORDBMS rather than an OODBMS or a RDBMS, because an ORDBMS would afford the extensibility promised by an OODBMS, while maintaining the efficiencies and standards of a RDBMS.

We initially implemented BRAID in Illustra; this initial implementation allowed us to define several image data types (e.g., binary, integer, RGB), and to support these data types with query-accessible image-processing and statistics operators. We subsequently used this database to manage and analyze data from several IBCTs [1]. Although Illustra supported our research, our reliance on a proprietary DBMS hindered our efforts to share our software and to extend and maintain BRAID. In particular, our goal of sharing our visualization software, statistical operators for data mining, and our Bayesian data-mining software was hindered by our reliance on a proprietary DBMS. Therefore, we opted to re-implement BRAID using open-source components.

In addition to sharing BRAID and its associated tools, we have developed data-mining software that analyzes neuroanatomic data; however, we did not integrate these tools into BRAID, due to concerns about computational efficiency. One example of such software is an application that we developed, which performs voxel-based lesion-deficit analysis. However, because this application is not called from BRAID's interface, our collaborators must retrieve data from BRAID, transform the data into a format acceptable to this analysis software, and then perform the analysis. This process is time-consuming and error-prone, and became more difficult as we developed more analysis applications. To facilitate the use of our data-mining software, we decided to integrate these tools into BRAID.

The rest of this paper is organized as follows. In Section 2, we describe the re-implementation of BRAID using open-source components, including the selection of a DBMS, design of a data model, and implementation of visualization operators. We describe our experience of integrating data-mining software into BRAID in Section 3. Section 4 provides examples of how to use BRAID to support image-based clinical trials. We conclude in Section 5.

DATABASE DESIGN

An additional, but by no means secondary, motivation that drove the reimplementation of BRAID, was our desire to overcome some deficiencies in the database design as implemented using Illustra. For example, we modeled clinical data by placing all data from all IBCTs into one table, where clinical variables were represented on a record-by-record basis: for each attribute-value pair, the attribute name was a text value, selected from four choices: "text", "integer", "real", or "Boolean". This design resulted in many NULL values in

this table, and caused inefficiencies during queries of, and modifications to, the database. In addition, this design made difficult the addition of a new IBCT to BRAID. To better model these data, we utilized an inheritance approach to represent the IBCTs included in BRAID. Another limitation of the previous implementation is that spatial operations were based only on the Talairach template space. Our reimplementation of BRAID allows each IBCT to choose its own coordinate system.

Our current implementation is based on PostgreSQL, a robust (and the only widely used) open-source ORDBMS (<http://www.postgresql.org>). Almost all functions of the structured query language (SQL) standard are implemented in the query language of PostgreSQL as a full-function relational database. Moreover, all object-oriented features are included as new specific instructions or reserved keywords. The server can handle simultaneous queries, users and databases. Several clients and interfaces for PostgreSQL are available; we have installed and used two interfaces, psql and pgaccess (<http://www.pgaccess.org>).

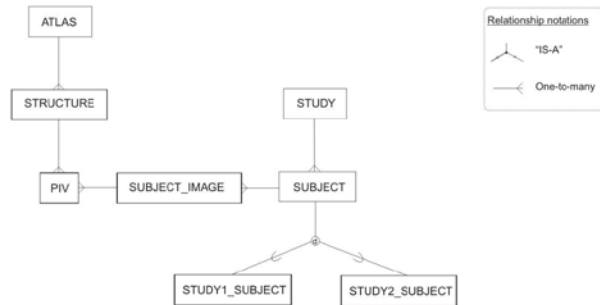
Software developers extend PostgreSQL by adding operators, usually written in C, to the system; database users and administrators can then query a database from within a program they have written in any of these languages. The PostgreSQL engine allows the user to develop new components, and to add them to the kernel as dynamically linked libraries. These components, represented as simple functions or procedures, may consist of SQL statements or C functions. To support IBCTs in BRAID, we defined an image data type, and image and statistical operators. Since we extended the database kernel using dynamically linked libraries, this process did not require recompilation of the kernel.

To facilitate sharing of BRAID's source code, we decided to use open-source components and development tools wherever possible. In addition to the DBMS, this environment includes the operating system, and the interface to the data. We chose the freely available hypertext preprocessor PHP (php: hypertext preprocessor, <http://www.php.net>) to generate BRAID's web pages that constitute the user-DBMS interface; as opposed to JavaScript, PHP is a server-side HTML-embedded scripting language, and is thus not client-dependent. We chose the open-source web server, Apache (<http://httpd.apache.org/>), to serve BRAID's web site. Finally, we chose to integrate these components under the Linux operating system (<http://www.linux.org>), and to compile extensions to BRAID using gcc (<http://www.gnu.org/gcc>). An important benefit of our choice of open-source components is the provision for any database administrator, through access to all source code, to construct and to maintain a brain-image database similar to ours.

Data model

Figure 1 presents an overview of BRAID's entity-relationship (E-R) diagram. The relationships between entities are either one-to-one or one-to-many. BRAID's database consists of data specific to each IBCT, in addition to a collection of anatomic atlases of reference brain-image volumes that describe the

Figure 1. The entity-relationship diagram for the BRAID schema. A rectangle represents an entity, and a line represents a relationship between entities. For the subclass/superclass (IS-A) relationship, the “d” notation specifies the constraint that subclasses must be disjoint; i.e., an entity can be a member of at most one of the subclasses.



spatial extents of brain structures. As shown in *Figure 1*, we model an atlas with two entities: ATLAS and STRUCTURE. The ATLAS entity encodes spatial information for each atlas included in BRAID [7]. The image of each atlas is color-coded to label different structures; that is, each atlas is an integer image. The STRUCTURE entity represents information related to atlas-defined brain structures. The main attribute of the STRUCTURE entity is *image*, which is defined as a binary mask in a particular coordinate space. The data for the STRUCTURE entity are extracted from color-coded atlas images.

Three entities—STUDY, SUBJECT, and SUBJECT_IMAGE—model subjects’ lesions, demographics, predisposing factors, and outcomes. The STUDY entity captures information for a particular study (e.g., study name), the SUBJECT entity records information relevant to each individual (e.g., age), and the SUBJECT_IMAGE entity captures information related to each subject’s image data. This entity includes an attribute of type *image* to refer to an image of a subject’s lesions. In addition, we include an additional subject entity for each IBCT, such as STUDY1_SUBJECT; this entity models clinical data specific to the corresponding IBCT, whereas the generic subject entity models subject data, such as date of entry, that we expect to maintain for all IBCTs. We model these distinct clinical variables by using the table-inheritance feature in PostgreSQL.

Finally, the PIV (precomputed intersection volume) entity connects the ATLAS and SUBJECT_IMAGE entities. PIV contains the volume of intersection of an atlas structure with a subject’s lesions; each intersection volume is computed off-line to improve the efficiency of queries.

Each of BRAID’s entities is mapped to a table in our extended relational-database implementation. As expected for a relational database, every row within a table is uniquely identifiable through the primary key for that table. We established relationships among entities by using primary and foreign keys that guarantee data integrity and coherence. In addition, we created derived fields to improve query performance; for example, we precomputed the volumes of

intersection in the precomputed-intersection-volume (PIV) table, because computing volumes for hundreds of atlas structures crossed with hundreds, or thousands, of subjects, is time consuming.

To map the subclass–superclass relationship onto tables, we represent the data for a study as a single table, in which columns represent clinical information and each row represents a subject. Each subject in a study has a unique identifier that acts as a key. To incorporate data from a new IBCT, a database administrator (DBA) need complete only four steps:

1. Create a new tuple for this study in the STUDY table.
2. Incorporate the basic information for the subjects of this new IBCT into the SUBJECT table, so that each subject has a unique number that is used to identify that subject’s clinical and image data.
3. Create a new table for this study’s clinical data, whose fields are the clinical variables of this particular IBCT.
4. Insert each subject’s image data into the SUBJECT_NUMBER table.

Data types

Unlike image databases that store only file names and locations of images, BRAID directly stores image data using our defined data type *image*. PostgreSQL maintains the *image* data type in binary form when queried, and thus can pass these data efficiently to data-processing programs, such as image-processing and statistical operators. Each defined type in PostgreSQL corresponds to a C-language structure. The key elements of the *image* data type are:

1. Modality (e.g., magnetic-resonance (MR), positron-emission tomography (PET))
2. Value (e.g., Boolean, integer, RGB)
 - We use the Boolean type for an image mask that represents, for example, a single atlas structure. We use the integer type to represent a subject’s lesion image, in which different integer values may be assigned to different lesions to distinguish them. We use the RGB type to represent an atlas image, in which each atlas structure is color-coded. We implemented this image value type as a C union, which facilitates extensibility.
3. Format (e.g. zyxx, raw)
 - This value indicates the format for the image data (see next element). For example, the zyxx format represents a line segment ($z, y, x_{\text{begin}}, x_{\text{end}}$) that has a certain value (e.g., color, signal intensity, 0/1 binary value).
4. Image data
 - The image data, encoded using the format specified in the previous element, are stored here.
5. Acquisition date
6. Voxel size in mm stored as three floats

Spatial operations

To support the execution of image-based queries, we defined and implemented a set of operators, including intersection, sum, and threshold, for the *image* data type. For example, we implemented the intersection procedure and mapped it to the “*” operator in PostgreSQL, which can be incorporated into any SQL statement. Similarly, we implemented the summation procedure and mapped it to the “+” operator.

In addition to implementing image operators, we have also implemented functions that take images as arguments; one example is the *threshold* function, which allows the user to apply different types of thresholds (integer or RGB values) to an image. The threshold function returns an *image* consisting of only those voxels satisfying the threshold criterion (all other voxels are set to zero). Another example is the *volume* function, which takes an image as its argument, and returns the volume of that image (i.e., corresponding to non-zero voxels) in mm³. We have also implemented functions for producing output: *save_image* and *generate_png*. The function *save_image* saves an image to a file, and the function *generate_png* generates a portable network graphics (PNG) image (<http://www.libpng.org/>) from BRAID’s internal image format.

Data security

We have implemented security primarily through access privileges, and de-identification of subjects’ data. To protect subjects’ privacy, our collaborating researchers strip all clinical and image data of potential identifiers, and give each subject a unique numerical ID, before sending the data to BRAID; this ID number, in conjunction with the study name, represents a key for that subject in the database. As an additional security measure we incorporate into BRAID only the subjects’ registered lesions, rather than the original MR images.

In addition, BRAID provides a user-level security model by assigning different levels of user access managed by the PostgreSQL server. BRAID’s database administrator (DBA) has full permission on all database objects and is responsible for all data-manipulation tasks: populating BRAID with new clinical or image data, and deleting or updating any datum. The DBA assigns for each IBCT a user or a group of users who will have read-only access (primarily through BRAID’s web site) to that study’s clinical and image data.

Data entry

BRAID provides a set of off-line C programs, run by BRAID’s DBA, that connect to the PostgreSQL server to populate the database with an IBCT’s image and clinical data. This process is facilitated by *psql* and other database-management tools available in PostgreSQL.

INTEGRATING ANALYTIC TOOLS INTO BRAID

After image preprocessing, segmentation of lesions, and registration to a common standard, the structural image data, which consist of registered lesions, are inserted into the database. For each subject, these image data, combined with clinical variables, constitute the data to be analyzed or visualized. With our assistance, our collaborators subsequently analyze the data, either testing a specific hypothesis or performing exploratory data mining. One way to perform this task is first to extract data from the database, and then use the standalone software to analyze these data. However, this process is time-consuming and error-prone.

To facilitate data mining for images and clinical data, we have begun to integrate BRAID’s off-line analytic tools into the ORDBMS. To support statistical analysis, we have linked BRAID to the open-source statistical computing package R (<http://www.r-project.org/>). Under this implementation, we use the ORDBMS to manage the data, to perform queries to select the subset of data for analysis, and to generate sufficient statistics; R takes as input these sufficient statistics to perform statistical analysis. We use the open-source package PL/R (<http://www.joeconway.com/plr/>) to support efficient communication between R and PostgreSQL.

To narrow down the vast array of statistical functions that R makes available to developers, we have chosen statistical and data-mining functions that support typical studies managed by BRAID. For example, BRAID was originally designed to support lesion-deficit analysis, in which an investigator’s goal is to determine associations among locations of abnormal voxels (or structures), and outcome measures or predisposing factors. Since voxels and outcome measures are often categorical, we chose to implement Fisher’s exact test.

Fisher’s exact test is computed based on a two by two contingency table. The contingency table counts binary values of (a) whether the state of a clinical variable for a subject meets a specific criterion; and (b) whether a voxel is labeled abnormal (i.e., lesion). This contingency table constitutes the sufficient statistics for Fisher’s exact test. That is, once we know the values in this contingency table, knowing the actual observations contributes no further information to statistical inference. In our implementation, PostgreSQL computes these sufficient statistics (i.e., the contingency table), and saves them temporarily as a table in the database. Because BRAID computes sufficient statistics interactively, BRAID supports exploratory data analysis, in which an investigator may frequently change the subset of subjects included in an analysis.

We implemented a PL/R function to analyze the contingency table generated from and transmitted by BRAID. In integrating this function into BRAID, we enable the ORDBMS to call R automatically to perform statistical inference, and then return the resulting p-value to BRAID’s user interface.

Integrating image-processing and statistical data-mining

Figure 2. An example of a visualization query performed in BRAID.

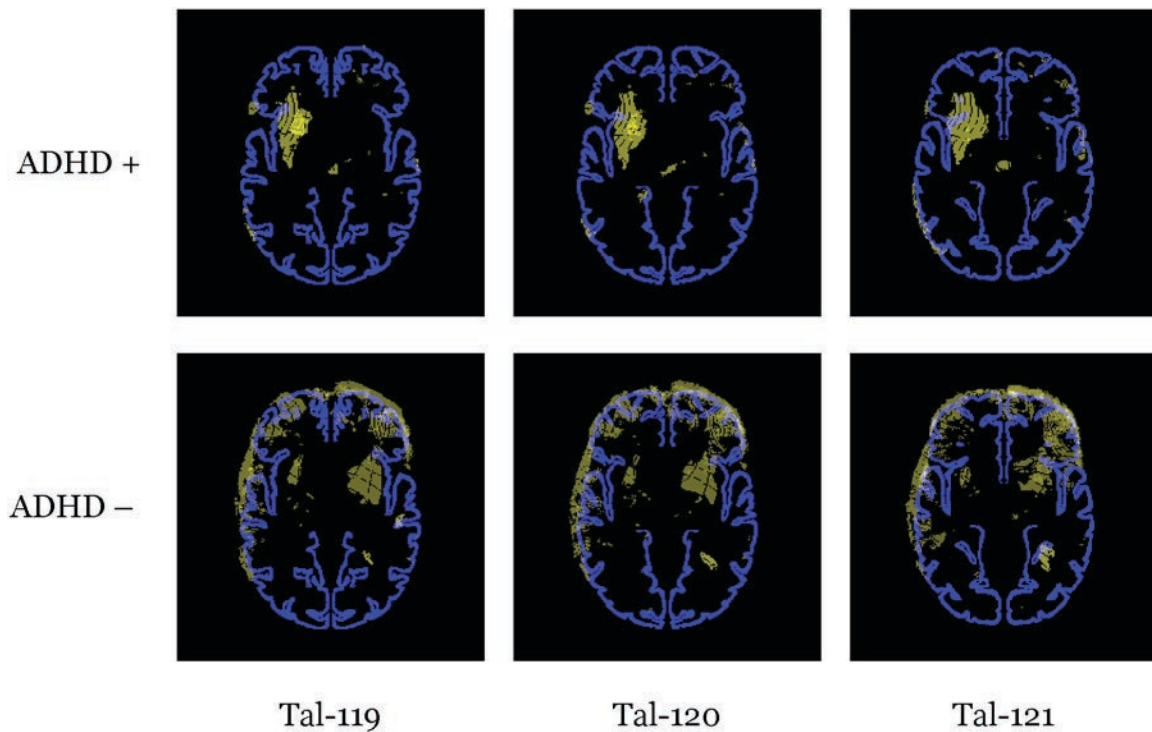


Figure 3. The main entry point of the web-based interface.

The PTBI study

Visualization

- [Subjects](#)
- [Queries](#)

Statistical Analyses

- Chi square test
 - [Chi squared analysis](#)
 - [A table of chi-squared results](#)
- Fisher exact test
 - [Fisher exact analysis](#)
 - [A table of fisher exact results](#)

Query by SQL

- [Example SQL queries](#)

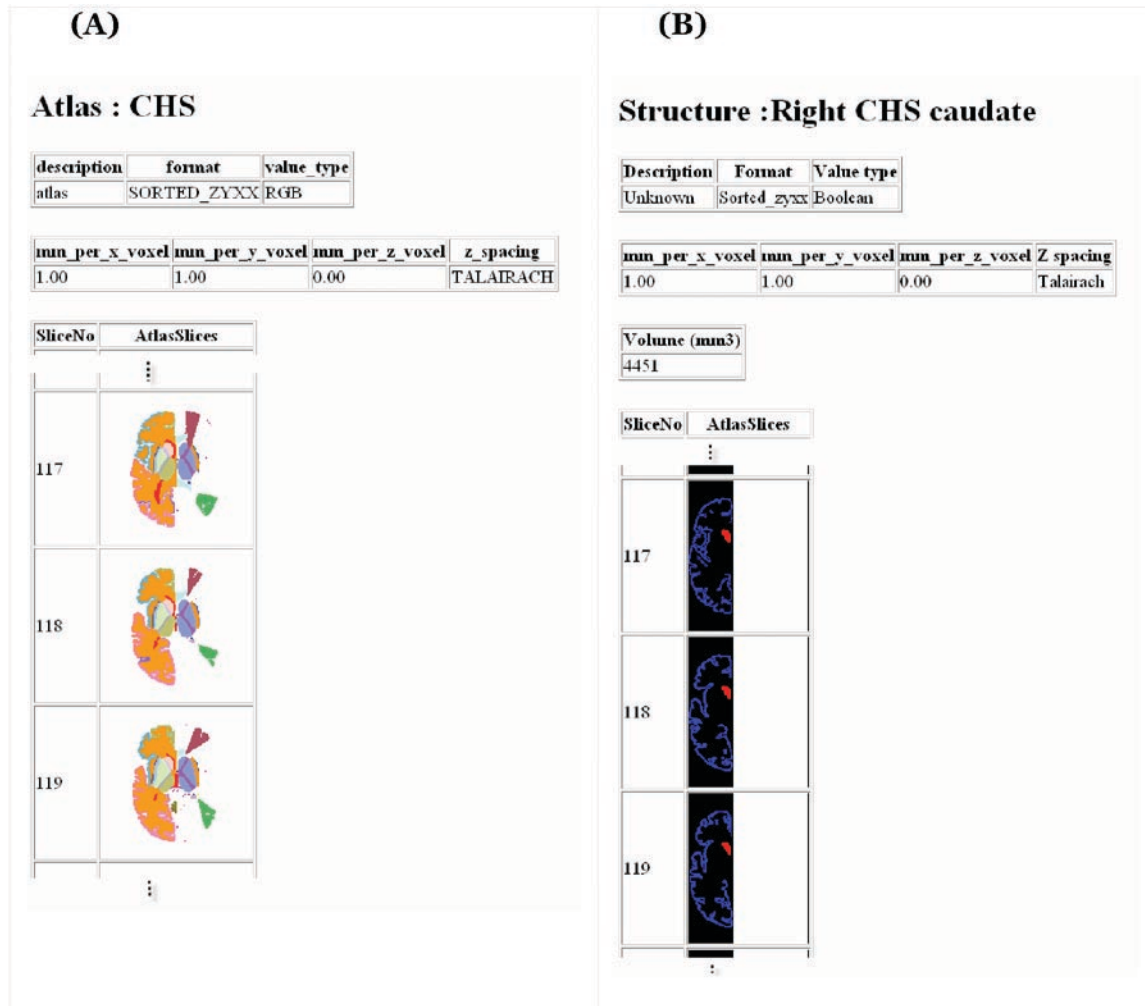
via the same interface. This feature frees our collaborators to focus on their research, rather than on the database. In turn, anyone implementing a statistical or data-mining function in R is free to change the details of that implementation, as long as the PL/R interface remains the same. Second, we use PostgreSQL to generate sufficient statistics; this process is not only efficient, but also safe because PostgreSQL has mechanisms for data integration. Third, developers benefit from the wide variety of statistical functions already implemented and validated in R.

USING BRAID

The following is an abbreviated outline of how BRAID is used to support IBCTs. When creating a new study, the collaborating principal investigator selects a coordinate system, which usually corresponds to an atlas, and a set of clinical data to store for each subject. The clinical collaborators send us their image and clinical data as they are collected; the image data contain regions of abnormal signal intensity, or lesions, which are delineated as regions of interest. We register all image data to the atlas originally selected for this study, and we ensure that all potential identifiers have been removed from the data. The database administrator incorporates these data into BRAID, after quality control in conjunction with clinical collaborators. We then assist our clinical collaborators in performing queries for visualization, statistical analysis, and data mining, using a web-based interface.

tools into BRAID has many benefits. First, users do not need to know about implementation details; they submit a request for analysis via BRAID's user interface, and they obtain the results

Figure 4. An example of visualizing atlas structures using BRAID's web interface. Query results are presented as series of axial PNG images. (A) The CHS atlas is shown as a color-coded composite of atlas structures. (B) The result of selecting a particular structure, the right caudate (shown in red), overlaid on Talairach cortex (shown in blue).



An example IBCT: the psychiatric sequelae of traumatic brain injury in children (PTBI)

As a motivating example of how BRAID assists in the management of IBCTs, we introduce a study of the psychiatric sequelae of traumatic brain injury in children (PTBI) [8]. The goal of this project is to investigate the effects of traumatic brain injury, by answering questions such as: Does the spatial pattern of brain lesions induced by traumatic brain injury predict the development of secondary attention-deficit hyperactivity disorder (S-ADHD)? Accordingly, working with MR brain images, the PTBI researchers delineate lesions that are related to traumatic brain injury, and then send us the original MR images, the segmented images, and the results of psychiatric and demographic data, stripped of any information that could uniquely identify any subject. We register each subject's MR images to a standard coordinate system, and use the resulting deformation field to register that subject's lesions.

Having incorporated these image and clinical data into BRAID, we can then assist PTBI researchers in performing

queries to determine whether the spatial distribution of lesions is associated with a particular clinical outcome. For example, PTBI researchers can generate a visualization query using BRAID's web-based interface, to examine the spatial distributions of lesions for subjects who developed S-ADHD, and for subjects who did not. These queries return axial summation images of all lesions across subjects who developed S-ADHD; a similar query returns the equivalent set of images for those subjects who did not develop S-ADHD. Figure 2 shows three representative axial images from these two visualization queries.

In addition to visual display of lesion locations across experimental groups, we can use BRAID to explore statistically associations between the spatial distributions of subjects' lesions and the existence of S-ADHD. For example, the input to BRAID's Fisher-exact aggregate function is a table, in which each row represents a subject, and which has two columns: S-ADHD status (normal/abnormal) and right-putamen status (abnormal if this structure's volume of intersection with a




Figure 5. Clinical information for a PTBI-study subject is shown in BRAID's web interface, followed by axial slices of the subject's brain lesions in yellow, overlaid on Talairach cortex in blue. The volumetric result is presented as series of axial PNG images. Only a few clinical data and axial image levels are shown for the sake of clarity.

PID	60
ACQUIREDDATE	2004-02-19
:	
ADHD_DEVEL	t
:	
:	

Description	Format	Value type
Subject all lesions	Sorted_zyxx	Integer

mm_per_x_voxel	mm_per_y_voxel	mm_per_z_voxel	Z spacing
1.00	1.00	0.00	Talairach

Number of Lesions	Volume (mm3)
23	48478

Slice No	Slice
:	
114	
115	
116	
:	

subject's lesions exceeds a user-supplied threshold). This function is invoked via the following SQL statement:

```
SELECT fisher_exact('PTBI', 'adhd_devel', 328, '=t', 0);
```

The aggregate function is called `fisher_exact`, and has 5 input values: the name of a study; the name of a binary clinical variable (`adhd_devel` indicates whether S-ADHD has developed); the index of an atlas structure (the right putamen in the Talairach atlas has index 328), the condition to be applied to the clinical variable (compare cases in which `adhd_devel` is true ['t'] to those in which it is not), and the lesion-volume threshold for labeling a structure abnormal for a particular subject (0 implies that we consider the right

putamen to be abnormal for any subject who has a lesion that overlaps with the right putamen by even one voxel). The result of this query is the p-value of the Fisher-exact test, along with the contingency table.

Clearly, it is unreasonable to expect clinical researchers to generate SQL statements. The following section describes BRAID's web-based user interface, which allows our collaborators and us to perform queries for visualization and for statistical analyses, such as those described above, without having to compose SQL statements.

USER INTERFACE

Query interface for visualization and statistical analysis

BRAID provides a simple graphical user interface (GUI) that allows users without extensive database experience to construct and submit complex visualization and statistical queries, and to browse the database, via a web browser. We implemented the interface to the database as a series of web pages that are dynamically generated by PHP.

Based on a user's access privileges, BRAID presents a list of studies, from which the user selects a study to browse; *Figure 3* shows an example of this type of page. At this point, the user may select any of the links shown, to browse a subject's lesion images, to build visualization query based on a combination of image and clinical criteria, to build a simple statistical query, or to view predefined examples of queries.

Browsing the database

As shown in *Figure 4*, BRAID allows users to view anatomic atlases. When a user requests to view an atlas, such as that from the Cardiovascular Health Study (CHS), BRAID also provides the user with a summary description of the atlas, based on voxel values in the corresponding *image* data type, followed by representative images that contain color-coded representations of various atlas structures, as shown in *Figure 4A*. BRAID also provides the user with a list of structures in that atlas. When a user selects a structure's hyperlink, the GUI sends the corresponding SQL query to the database. As shown in *Figure 4B*, BRAID returns a series of axial images demonstrating the selected structure superimposed on Talairach cortex, which is optionally provided as a spatial reference.

BRAID's GUI also provides the user the ability to view the data for any subject in the database to which they have been granted access, by selecting from a list of subjects. BRAID's GUI dynamically generates a SQL statement, which in turn is sent to the PostgreSQL server. As shown in *Figure 5*, the result of this query is a table of the subject's clinical information, followed by the subject's images. For example, *Figure 5* shows three of the clinical variables in the PTBI study: `pID` (a unique patient identifier), `acquired-date` (the date that these data were collected) and the Boolean variable `adhd_devel`, which indicates whether that subject developed S-ADHD, assessed 1 year after injury.

Figure 6. (A) BRAID's web interface for constructing a visualization query. The user selects from menus of anatomical structures and clinical variables, and specifies a subject-selection criterion. The user can color each structure, and lesions, using a palette. (B) The SQL statement generated by BRAID is displayed; a user comfortable with SQL can modify the query if desired, before submitting it to BRAID. The images on the right are a subset of those generated by the SQL statement on the left.

(A)

Structures
Right Talairach putamen (7.41950e+3) #FF0000
Left CHS thalamus (9.34250e+3) #00FFFF
☒ Show Talairach Cortex #0000FF
Add Structure Fewer

Variables
Lesions Color: #FFFF00
ADHD_DEVEL (type:bool, range: 0 => 1) =t
Add Selection Criterion Fewer ☒ Match All ☐ Match Any
Reset Build Query

(B)

```

SELECT(
  map_image(
    (SELECT img FROM Structures WHERE side='Left'
      and atlas='Talairach' and name='cortex') +
    (SELECT img FROM Structures WHERE side='Right'
      and atlas='Talairach' and name='cortex'),
    '#0000FF') +
  map_image(
    (SELECT img FROM Structures WHERE side='Right'
      and atlas='Talairach' and name='putamen'),
    '#FF0000') +
  map_image(
    (SELECT img FROM Structures WHERE side='Left'
      and atlas='CHS' and name='thalamus'),
    '#00FFFF') +
  map_image(
    sum_study_lesions('PTBI', '(Subjects.adhd_devel="t")'),
    '#FFFF00')
)::PNG

```

Slice No	Slice
115	
116	
117	
118	
119	

Visualization queries

The visualization-query form (Figure 6A) allows a user to display lesions for a group of subjects; BRAID then sums binary lesion maps at each voxel location to return a 3-dimensional integer map. First, the user selects optional atlas structures from a menu; these structures will be superimposed on the lesion map to provide spatial context for visualization. The user can choose a color for each selected structure from a web palette. A checkbox allows the user to superimpose Talairach

cortex on these structures and/or lesions, since this structure is the one most commonly chosen in this setting. Note that structures can be mixed from different atlases, which is one of the advantages of requiring that image data be registered to a common standard.

Next, the user chooses a color for displaying lesions; as with atlas structures, the default color for lesions can be modified easily using the supplied palette. Finally, the user can choose to include all subjects in a particular study, or to

apply selection criteria, such as including only subjects who developed S-ADHD (e.g., `adhd_devel = 't'` in Figure 6A); additional criteria can be added for more complex queries.

After the user has selected anatomic structures, colors, and subject-selection criteria, and submitted this form to the database, BRAID builds the corresponding SQL query dynamically, and displays it to the user for optional editing before submission to PostgreSQL. In this way, BRAID provides users who are facile with SQL the option of modifying the SQL statement before submitting it. Once the SQL statement is submitted over the web to the database, BRAID will return images showing the summation of subjects' lesions superimposed on the selected structures, as shown in Figure 6B.

The corresponding SQL statement in Figure 6B demonstrates how we use the schema and spatial operators. As expected, we use the '+' operator for voxel-wise summation. Each voxel in the resulting image volume corresponds to the number of subjects for whom this voxel is abnormal. The `map_image` procedure maps images to a color, and the `sum_study_lesions` procedure tallies the numbers of lesions across subjects of a particular study who meet the specified selection criteria. To show the result of this query on a web browser, BRAID converts the image volume into a series of axial PNG images by simple typecasting, using the "::PNG" operator.

Figure 7. BRAID's web interface for performing Fisher-exact analysis. (A) The user selects an atlas structure and a clinical variable, and enters a lesion-volume threshold, since Fisher-exact analysis requires binary variables. (B) The p-value (≈ 0.02) and the 2 x 2 table (L = lesion, D = deficit) are returned.

(A) Choose a structure

Right Talairach POM/hypothal (5.40000e+1)
Right Talairach post/hypothal (2.80000e+1)
Right Talairach p.thal=pulvinar (2.27000e+3)
Right Talairach putamen (7.41950e+3)
Right Talairach PV/hypothal (2.40000e+1)
Right Talairach pyr1/face (1.08300e+3)
Right Talairach pyr2/arm (2.31900e+3)
Right Talairach pyr3/legs/trnk (1.60000e+3)

Choose an attribute

ADHD_DEVEL (type:bool, range: 0 => 1)
CS2P (type:bool, range: 0 => 1)
DX7P2 (type:bool, range: 0 => 1)
ENUR2P (type:bool, range: 0 => 1)
PTSDX2C (type:bool, range: 0 => 1)
PTSDX2P (type:bool, range: 0 => 1)
PTSDGRPC (type:bool, range: 0 => 1)
PTSDGRPP (type:bool, range: 0 => 1)

Threshold Values

Enter minimum volume for intersection 0

Continue >

(B) Fisher exact analysis results

structure : Right Talairach putamen (minvol=0)

varname	structure	p	N	L	D	L-D+	L-D-	L-D+
ADHD_DEVEL	Right Talairach putamen	0.021708	76	56	10	5	5	

Statistical queries

BRAID also provides a web-based interface to basic statistical-analysis procedures. Figure 7A shows an example of how a user would perform Fisher-exact analysis. First, the user selects an atlas structure, such as right Talairach putamen, from the menu. Second, the user selects a clinical variable for structure-function analysis; in this example, the user is interested in determining whether the development of ADHD is associated with the presence or absence of lesions in the right putamen (as defined in the Talairach atlas). Because lesion volume is a continuous variable, and Fisher exact analysis requires binary variables, we provide a field in which the user can enter a threshold, which converts lesion volume into a binary (normal/abnormal) variable. In this case, if there is any overlap (> 0 voxels), the structure is labeled abnormal for the purposes of Fisher-exact analysis. The user submits this form to BRAID over the web, and BRAID returns the result in a tabular format (Figure 7B).

CONCLUSIONS AND FUTURE WORK

We have described the design, open-source implementation, and use of BRAID, a system for managing, querying, analyzing and visualizing brain MR images. We implemented BRAID using an open-source object-relational DBMS, PostgreSQL, extended to handle spatial data types and user-defined functions. To facilitate sharing of BRAID's software tools, we built BRAID using open-source components such as PostgreSQL, Apache and PHP, running on the Linux operating system. BRAID is available for download from its web site (<http://braid.uphs.upenn.edu>), supported by documentation for installing and running BRAID.

We identified and analyzed requirements for supporting image-based clinical trials. We introduced and implemented the *image* data type, along with image-processing and statistical operations within PostgreSQL. We have extended the *image* data type to include lesion type, to facilitate lesion-deficit analysis for different types of lesions, such as computed tomography, MR diffusion, MR perfusion, etc.

We implemented, and integrated into BRAID, statistical tests to support exploratory data-mining queries such as "compute Fisher-exact p-values for the pair-wise combination of brain structures and clinical conditions" for a particular study. By linking the open-source statistics package R to BRAID, we have greatly expanded the range of on-line analyses available to our collaborators from BRAID's web interface.

We plan to integrate data-mining software based on Bayesian networks into BRAID. In our experience, these methods are computationally tractable, and can detect nonlinear multivariate structure-function associations. We can accomplish this task using our source code, or the R package called Deal, which supports generating Bayesian-network models from data.

ACKNOWLEDGMENTS

This work was supported by National Institutes of Health grant R01 AG13743, which is funded by the National Institute of Aging, and the National Institute of Mental Health.

REFERENCES

1. Herskovits EH. An architecture for a brain-image database. *Meth Inform Med*. 2000. 39(4-5): 291–7.
2. Herskovits EH, Megalooikonomou V, Davatzikos C, Chen A, Bryan RN, Gerring JP. Is the spatial distribution of brain lesions associated with closed-head injury predictive of subsequent development of attention-deficit hyperactivity disorder? Analysis with a brain-image database. *Radiol*. 1999. 213(2):389–94.
3. Shen D, Davatzikos C. HAMMER: Hierarchical attribute matching mechanism for elastic registration. *IEEE Trans Med Imaging*. 2002. 21(11):1421–39.
4. Elmasri R, Navathe S. *Fundamentals of Database Systems*. 3rd ed. Addison Wesley; 1999. 1009 p
5. Diallo B, Traverre JM, and Mazoyer B. A review of database management systems suitable for neuroimaging. *Meth Inform Med*. 1999. 38(2):132–9.
6. Stonebraker M. *Object-relational DBMSs: The Next Great Wave*. San Francisco: Morgan Kaufmann; 1996. 216 p
7. Talairach J , Tournoux P. *Co-planar Stereotaxic Atlas of the Human Brain*. Thieme Medical Publishers. 1988. 122 p
8. Gerring J, Brady K, Chen A, Herskovits E, Bandeen-Roche K, Denckla MB, Bryan RN. Premorbid prevalence of ADHD and development of secondary ADHD after closed head injury. *J Am Acad Child Adolesc Psychiatry*. 1998. 37(6):647–54.