An Object Relational Approach to Biomedical Database

Phillip C-Y Sheu, Brian Cummings, Carl Cotman, Charlie Chubb
and the BioCompose Group:
L. Hu, T. Wang, Julene Johnson, Scott Mobley, Tom Sitch, Y. Inagaki

Institute of Brain Aging and Dementia
University of California
Irvine, CA 92697

Email contact: sheu@ece.uci.edu

Abstract

As neuroscience research advances, there is increasing need to integrate data from a single patient who has undergone multiple levels of analysis. In a biomedical application (such as an Alzheimer's clinic) a patient might have medical data, MRIs, cognitive data and autopsy data all collected at various times. This paper introduces “BioCompose”, an object relational database system that integrates clinical, neuropathological and image data sets into a uniform framework with a simple user interface. It addresses the advantages of taking an object-relational approach for a large and complex application.

I. INTRODUCTION

Although the degenerative pathology in AD is often described as diffuse, it is well known that this pathology is not uniformly distributed within the brain, especially at different stages of the disease. Numerous studies have shown that certain regions of the brain are more vulnerable to neurodegeneration than others. This regional specificity is derived from selective vulnerability and plasticity at the cellular level. Because different domains of pathological cognitive functioning rely on the integrity of specific neural structures, the patterns of regional specificity also have consequences on cognition and behavior. That is, this regional specificity is evident in patterns of selective impairments and preservations of cognitive abilities. For example, some patients show a disproportionate impairment on tests of frontal lobe function (e.g. planning). It is possible that some of the patients who exhibit “selective cognitive impairments” are indeed variants of AD. However, others may be exhibiting an entirely unique pathology.

In order to systematically investigate such observations from the clinical level to the cellular level and back again, we have developed “BioCompose”, an object relational database system that integrates clinical, neuropathological and image data sets into a uniform framework with a simple user interface. It addresses the advantages of taking an object-relational approach for a large and complex application.

II. RELATED WORK
In the past, most research in biomedical databases has focused on biomedical image management (see, e.g., [2]). More recently, efforts have been made in applying advanced data modeling and database techniques to store other types of biomedical data. For example, in [4], an object-oriented data storage system was employed to store cellular data and facilitate index-based retrieval of such data. In [5], a querying mechanism was proposed to retrieve biomedical data based on the concept of entity-attribute-value. In [6], a suite of tools, which allow users to compose queries against a medical database without knowing database terminology, was reported.

To our knowledge, BioCompose may be the first database that integrates biomedical data at different levels of abstraction (i.e., cellular variables, pathological diagnosis, medical information etc.) that supports the expressive power of object relational queries and object relational data analysis. On the other hand, although the object relational framework has shown significant promise as the next-generation database system, there has been a lack of evidence that it is superior to the relational framework for a large and complex application.

III. CONCEPTS

This section outlines the design objectives and the rationale behind the object relational approach taken by BioCompose.

III.1 Drawbacks of Relational Databases

Most of today’s clinical databases have been organized in tables on a relational database server. This approach, however, has several drawbacks at least in theory. The first is that relational databases were not designed for, and typically do not have the capacity to support, multi-media information.

Another limitation with a relational system is data retrieval. The standard query language for relational system, SQL, is limited in scope (i.e., some queries cannot be expressed in SQL; e.g., finding patients similar in “behavior”.) On the other hand, in a relational database system information has to be stored as tables (relations). This is not a natural representation for complex objects such as a “patient”, which consists of initial patient intake data, his/her medical history, physical exams, neuropsychological tests, neuropathologic data, reports, laboratory results, neuroimaging, diagnosis, and treatments. Many of these components, likewise, contain a hierarchical structure within them. For example, for neuropathologic data, the lowest level object that can be modeled is a cell, whose attributes include shape, color, and geometric location. Above the cells there are sections and regions. If different components of a brain are stored into different tables, the user would have to assemble them back to a logical structure in order to answer meaningful questions.

III.2 BioCompose Overview
Figure 1. In this example of a Reporting Tool, the user has already selected a case (002-97) using a Web browser and requested to view anatomical data on the presence of neuropathology within different brain regions. On the upper-left Web page, the user has selected “Slice 7” as containing the brain region of interest. The browser changes (upper-right) to show a line drawing and grayscale image of a coronal slice through this region of the brain. Next, by clicking on “Go to Tissue” the user requests to see the gross image of the selected case (lower-right) and associated high-level information (age, sex, diagnosis) associated with this case. From the gross image, the user can select even more low-level images and view them on-line (lower-left).

An object relational database such as BioCompose can provide a seamless integration of clinical, pathological and cellular information presented in various forms with the uniform concept of object. It allows the user to access a logical object as a set of hierarchically structured records without worrying about how they are stored physically. That is, from a patient object the user can access any variable following its logical structure. It allows users to pose high level queries that are not feasible in a relational database (e.g., find similar). The framework is also designed to allow advanced pattern recognition capabilities and knowledge discovery functions to be added incrementally as it evolves. Consequently, it is a completely open framework whose power and intelligence can expand with time, and as the user’s needs grow.

On top of the object structure, the BioCompose system provides a query tool, a reporting tool, and an analysis tool:
Query Tool: Queries in *BioCompose* are structured along the lines of natural language and sentences. Objects (nouns) are identified, described (adjectives, predicates) and acted upon (verbs). Queries are composed by naive users based on simple multiple hierarchical choices without knowing any low-level concepts such as “join” and “selection”. Suppose, for example, that an investigator wishes to identify the number of patients with a clinical diagnosis of Alzheimer's disease who have diabetes and have been generated for a specific genetic marker (e.g. ApoE ) genotype results available. The query would consist of a noun (*patients*), three predicates (with clinical diagnosis of AD, having diabetes, having ApoE genotype results available) and a verb (find).

![Image](image_url)

*Figure 2.* Analysis tool shows a patient “Neuropsychological Profile” which has been plotted “on-the-fly” from data in various tables. In this example, we can see that the patient in question is primarily impaired on Frontal tasks.

Reporting Tool: The reporting tool allows the user to navigate an object in a hierarchical fashion. In particular, from a high-level object (e.g. patient) the user can “drill down” to lower-level objects that comprise the higher-level object; similarly, from a low-level object the investigator can “roll up” to higher-level objects. Figure 1 shows one style of “drilling down” through data from a human brain.

Analysis Tool: The analysis tool aggregates low-level data and provides summary information. In this way the user can gather data from multiple tests and view them graphically on one form, or request aggregate data on a group of patients. In Figure 2, a clinician is requesting BioCompose to plot the “Neuropsych” profile on a selected patient to see how far from the norm his performance is on a wide range of different neuropsychological tests.
IV. IMPLEMENTATION

This section summarizes the object structure, vocabulary and query structure in BioCompose.

IV. 1 Object Schema

The basic "facts" about image data in BioCompose can be stored according in the relational form as follows.

```
create table ColorSet  
  (ColorSetID : string : 4, 
   EvaluationID : string : 4, 
   PriorProbability : float : 8, 
   CovarianceMatrix_1 : float : 8, 
   CovarianceMatrix_2 : float : 8, 
   CovarianceMatrix_3 : float : 8, 
   CovarianceMatrix_4 : float : 8, 
   CovarianceMatrix_5 : float : 8, 
   CovarianceMatrix_6 : float : 8, 
   Class_Pixelcount : int : 4) key : ColorSetID
```

```
create table Evaluation  
  (EvaluationID : string : 4, 
   ImageID : string : 4, 
   MinSize : int : 4, 
   FilterRepet : int : 4, 
   EvaluatedBy : string : 255, 
   EvaluatedDate : string : 20, 
   Cellversion : string : 255, 
   Signature : string : 255, 
   Class1_Pixels : int : 4, 
   Class2_Pixels : int : 4, 
   Background_Pixels : int : 4, 
   Knowledge_Base : string: 128) key : EvaluationID
```

```
create table Image  
  (ImageID : string : 4, 
   ImageDate : string : 20, 
   ImageSource : string : 50, 
   CaseID : string : 32, 
   CaseSource : string : 50, 
   CaseNumber : int : 4, 
   CaseYear : int : 4, 
   SlideSource : string : 50, 
   Slide# : string : 32, 
   CameraID : int : 4, 
   Camera_Calibration : string : 2, 
   MicroscopeID : int : 4, 
   Objective : int : 4, 
   Photo_eyepiece : int : 4, 
   Numerical_Aperture : float : 8, 
   Filter1 : int : 4, 
   Filter2 : int : 4, 
   Filter3 : int : 4, 
   Illumination_Voltage : float : 8, 
   Light_path : int : 4, 
   Final_Magnification : int : 4, 
   X_coordinate : float : 8, 
   Y_coordinate : float : 8, 
   Z_coordinate: float : 8, 
   Image_File_Path Char : string: 255) key : ImageID
```

```
create table Object  
  (ObjectID : string : 4, 
   ObjectClusterID : string : 4,
```

5
select count(s.objectID) from Object s, ObjectCluster t, Evaluation v where s.ObjectClusterID = t.ObjectClusterID and t.EvaluationID = “001” and v.EvaluationID=t.EvaluationID and v.ImageID = “0001”

• [Q2] List objects of all clusters of all evaluations of each image.

The “star” schema, which is completely bottom-up, makes many high-level queries difficult to express. Following are some obvious examples that are expressed in SQL.
For a database of 580 images, 30 evaluations, 161 object clusters and 306 objects, the execution time of [Q1] is 4 seconds and the execution time of [Q2] is 49 seconds. As will be shown in Section IV.2, [Q1] and [Q2] can be expressed in a much simpler way (and with better performance) if we reconstruct the above schema in an object relational form, where several attributes (underlined) are added to express aggregations of objects. For example, the attribute “Evaluations” is added to store the set of evaluations - in terms of their unique identifiers - that have been done for a particular image.) as illustrated below.

```sql
create table Image (  
  ImageID : string : 4,  
  ImageDate : string : 20,  
  ImageSource : string : 50,  
  CaseID : string : 32,  
  CaseSource : string : 50,  
  CaseNumber : int : 4,  
  CaseYear : int : 4,  
  SlideSource : string : 50,  
  Slide# : string : 32,  
  CameraID : int : 4,  
  Camera_Calibration : string : 2,  
  MicroscopeID : int : 4,  
  Objective : int : 4,  
  Photo_eyepiece : int : 4,  
  Numerical_Aperture : float : 8,  
  Filter1 : int : 4,  
  Filter2 : int : 4,  
  Filter3 : int : 4,  
  Illumination_Voltage : float : 8,  
  Light_path : int : 4,  
  Final_Magnification : int : 4,  
  X_coordinate : float : 8,  
  Y_coordinate : float : 8,  
  Z_coordinate : float : 8,  
  Evaluations:setof_Evaluation,  
  Image_File_Path Char : string: 255) key : ImageID
```

```sql
create table Evaluation (  
  EvaluationID : string : 4,  
  ImageID : Image : 4,  
  MinSize : int : 4,  
  FilterRepet : int : 4,  
  EvaluatedBy : string : 255,  
  EvaluatedDate : string : 20,  
  Cellversion : string : 255,  
  Signature : string : 255,  
  ColorSets:setof_ColorSet,  
  ObjectClusters:setof_ObjectCluster,  
  Class1_Pixels : int : 4,  
  Class2_Pixels : int : 4,  
  Background_Pixels : int : 4,  
  Knowledge_Base : string: 128) key : EvaluationID
```

```sql
create table ColorSet (  
  ColorSetID : string : 4,  
  EvaluationID : string : 4,  
  PriorProbability : float : 8,  
  CovarianceMatrix_1 : float : 8,  
  CovarianceMatrix_2 : float : 8,  
  CovarianceMatrix_3 : float : 8,  
  CovarianceMatrix_4 : float : 8,
```
The object schema at the clinical level is shown in Figure 3.
IV. 2 Object Vocabulary and Queries

As noted earlier, one of the major advantages of BioCompose is the expressive power of its queries. Queries in BioCompose are composed from three subtypes of “terms”: nouns, adjectives, and verbs. Queries can also contain mathematical operators (i.e. less than, more than, equal). The following table summarizes some basic terms at clinical, pathological, and image levels. As described earlier, terms can be added to BioCompose dynamically.

A basic query sentence in BioCompose is of the form:

(Verb) those (Nouns) which are (condition) and (condition) and …

where condition is presented in one of the following two forms:

which are (qualifier)
whose (quantifier) are (mathematical operation)(comparison value)

As an example, the two queries discussed in Section IV.1 can be expressed in BioCompose as (given a particular image and an evaluation):

[Q1]
\[
\begin{align*}
\text{Count}_\text{object}(s.\text{ObjectClusters}) \\
\text{from } \text{Evaluation } s \\
\text{where } s.\text{EvaluationID} = '001' \text{ and } s.\text{ImageID} = '0001'
\end{align*}
\]

[Q2]
\[
\begin{align*}
\text{show}_\text{Image}(s) \\
\text{from } \text{Image } s
\end{align*}
\]

The execution time for [Q1] is 2 seconds and the execution time for [Q2] is 23 seconds. The following are some additional queries that can be posted in BioCompose:

- [Q3] Find neurons which are pyramidal.
- [Q4] Find glia which are microglia and which are reactive.
- [Q5] Find tangles which are intra-cellular and whose diffusing factor are greater than 1.2.
- [Q6] Find plaques whose number of neurites is less than 20.
- [Q7] Find images which have plaques whose number of neurites is less than 20.
- [Q8] Find images whose diffuse plaque number is greater than neuritic plaque number.

<table>
<thead>
<tr>
<th>Verb</th>
<th>Noun</th>
<th>Qualifier</th>
<th>Dependent-Qualifier</th>
<th>Quantifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count elements of Set (Cluster)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Find</td>
<td>Count</td>
<td>Classify</td>
<td>Show</td>
<td></td>
</tr>
<tr>
<td>(Image) Object</td>
<td>(Image) Object</td>
<td>(Image) Object</td>
<td>(Image) Object</td>
<td></td>
</tr>
<tr>
<td>Neuron</td>
<td>Pyramidal</td>
<td>Stellate</td>
<td>Granule</td>
<td>Bipolar</td>
</tr>
<tr>
<td>Glia</td>
<td>Astrocyte</td>
<td>Microglia</td>
<td>Oligodendroglia</td>
<td>Schwan Cell</td>
</tr>
<tr>
<td></td>
<td>Reactive</td>
<td>Non-reactive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tangle</td>
<td>Pretangle</td>
<td>Intra-cellular</td>
<td>Extra-cellular</td>
<td>Ghost Tangle</td>
</tr>
<tr>
<td>Amyloid</td>
<td>Small</td>
<td>Large</td>
<td>Diffused Clouded</td>
<td></td>
</tr>
<tr>
<td>Plaque</td>
<td>Diffuse</td>
<td>Primitive</td>
<td>Neurite</td>
<td>Burnt-out</td>
</tr>
<tr>
<td>Synapse</td>
<td>Pre-synaptic</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Verb | Noun | Qualifier | Dependent-Qualifier | Quantifier       |
---   |      |           |                     |                  |
Count elements of Set (Cluster) |                     |                      |                     |                  |
Find  | Count | Classify | Show                |                  |
(Image) Object | (Image) Object | (Image) Object | (Image) Object |                  |
Neuron | Pyramidal | Stellate | Granule | Bipolar |
Glia | Astrocyte | Microglia | Oligodendroglia | Schwan Cell |
             | Reactive | Non-reactive |                     |                  |
Tangle | Pretangle | Intra-cellular | Extra-cellular | Ghost Tangle |
Amyloid | Small | Large | Diffused Clouded |                  |
Plaque | Diffuse | Primitive | Neurite | Burnt-out |
Synapse | Pre-synaptic |              |             |            |
<table>
<thead>
<tr>
<th>Patient</th>
<th>Post-synaptic Axoaxonic</th>
<th>Size Uniformity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Demented</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mild Dementia</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Moderate Dementia</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Severe Dementia</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Non-Demented</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cognitive Domains Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Frontal Lobe Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Language Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Attention/Concentration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Memory Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cognitive Skill Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Object Naming Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Visual Memory Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Copying Ability Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Functional Skill Impaired</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Driving Skill Impaired</td>
<td></td>
</tr>
</tbody>
</table>

The above qualifiers can be realized as predicates (methods) in BioCompose whose values are determined at query execution time. In a relational database we have to pre-compute the values of these predicates and store them as tables such as following:

```sql
CREATE TABLE Pyramidal(
    ObjectID: string: 4) key: ObjectID

CREATE TABLE Glia(
    ObjectID: string: 4) key: ObjectID

CREATE TABLE Microglia(
    ObjectID: string: 4) key: ObjectID

CREATE TABLE Reactive(
    ObjectID: string: 4) key: ObjectID

CREATE TABLE Tangle(
    ObjectID: string: 4) key: ObjectID

CREATE TABLE Intra_cellular(
    ObjectID: string: 4,
    diffuse_factor: float: 8) key: ObjectID

CREATE TABLE Plaque(
    ObjectID: string: 4,
    Num_neu: int: 4,
    load_occupy: float: 8) key: ObjectID

CREATE TABLE Diffuse(
    ObjectID: string: 4) key: ObjectID

CREATE TABLE Neurific(
    ObjectID: string: 4) key: ObjectID
```

Now [Q3] – [Q7] can be expressed in SQL as follows:

```sql
SELECT w.ObjectID
FROM Object s, Image t, Evaluation u, ObjectCluster v,
```
Neuron \(w\), Pyramidal \(k\)
where \(s.\)ObjectClusterID = \(v.\)ObjectClusterID and 
v.EvaluationID = \(u.\)EvaluationID and 
\(u.\)ImageID = \(t.\)ImageID and \(s.\)ObjectID=\(w.\)ObjectID 
and \(u.\)EvaluationID = '001' 
and \(w.\)ObjectID = \(k.\)ObjectID and \(t.\)ImageID='0001' 

\[Q4\]
select \(w.\)ObjectID 
from Object \(s\), Image \(t\), Evaluation \(u\), ObjectClass \(v\), 
Glia \(w\), MicroGlia \(k\), Reactive \(x\) 
where \(s.\)ObjectClusterID = \(v.\)ObjectClusterID and 
v.EvaluationID = \(u.\)EvaluationID and 
\(u.\)ImageID = \(t.\)ImageID and \(s.\)ObjectID=\(w.\)ObjectID 
and \(w.\)ObjectID = \(k.\)ObjectID and \(w.\)ObjectID=x.\)ObjectID 
and \(u.\)EvaluationID = '001' 
and \(t.\)ImageID = '0001' 

\[Q5\]
select \(w.\)ObjectID 
from Object \(s\), Image \(t\), Evaluation \(u\), ObjectCluster \(v\), 
Tangle \(w\), Intra_cellular \(k\) 
where \(s.\)ObjectClusterID = \(v.\)ObjectClusterID and 
v.EvaluationID = \(u.\)EvaluationID and 
\(u.\)ImageID = \(t.\)ImageID and \(s.\)ObjectID=\(w.\)ObjectID 
and \(w.\)ObjectID = \(k.\)ObjectID and 
\(k.\)diffuse_factor > 1.2 and \(t.\)ImageID='0001' 

\[Q6\]
select \(t.\)ObjectID 
from Object \(s\), Image \(t\), Evaluation \(u\), ObjectCluster \(v\), 
Plaque \(w\) 
where \(s.\)ObjectClusterID = \(v.\)ObjectClusterID and 
v.EvaluationID = \(u.\)EvaluationID and 
\(u.\)ImageID = \(t.\)ImageID and \(s.\)ObjectID=\(w.\)ObjectID 
and \(w.\)ObjectID < 20 and \(t.\)ImageID='0001' 

\[Q7\]
select \(s.\)ObjectID 
from Object \(s\), Image \(t\), Evaluation \(u\), ObjectCluster \(v\), 
Plaque \(w\) 
where \(s.\)ObjectClusterID = \(v.\)ObjectClusterID and 
v.EvaluationID = \(u.\)EvaluationID and 
\(u.\)ImageID = \(t.\)ImageID and \(s.\)ObjectID=\(w.\)ObjectID 
and \(w.\)Num_neu < 20 and \(u.\)EvaluationID = '001' 
and \(t.\)ImageID='0001' 

\[Q8\]
select \(t.\)CaseID 
from Image \(t\) 
where (select count(\(w.\)ObjectID) 
from Object \(s\), Evaluation \(u\), ObjectClass \(v\), 
Plaque \(w\), Diffuse \(k\) 
where \(s.\)ObjectClusterID = \(v.\)ObjectClusterID and 
v.EvaluationID = \(u.\)EvaluationID and 
\(u.\)ImageID = \(t.\)ImageID and \(s.\)ObjectID=\(w.\)ObjectID 
and \(w.\)ObjectID = \(k.\)ObjectID) >
On the other hand [Q3] – [Q8] can be expressed as object relational queries as follows:

[Q3]
```
select t.ObjectID
from Object t
where neuron(t) and pyramidal(t) and
  t.ObjectClusterID.EvaluationID = '001' and
  t.ObjectClusterID.EvaluationID.ImageID = '0001'
```

[Q4]
```
select t.ObjectID
from Object t
where glia(t) and microglia(t) and reactive(t) and
  t.ObjectClusterID.EvaluationID = '001' and
  t.ObjectClusterID.EvaluationID.ImageID = '0001'
```

[Q5]
```
select t.ObjectID
from Object t, float s
where tangle(t) and intra_cellular(t,s) and s > 1.2
  and t.ObjectClusterID.EvaluationID = '001' and
  t.ObjectClusterID.EvaluationID.ImageID = '0001'
```

[Q6]
```
select t.ObjectID
from Object t, int s
where plaque(t,s) and s < 20
  and t.ObjectClusterID.EvaluationID = '001' and
  t.ObjectClusterID.EvaluationID.ImageID = '0001'
```

[Q7]
```
select t.ObjectID
from Object t, int s
where plaque(t,s) and s < 20
  and t.ObjectClusterID.EvaluationID = '001' and
  t.ObjectClusterID.EvaluationID.ImageID = '0001'
```

[Q8]
```
select t.CaseID
from Image t
where count_diffuse_plaque(t) > count_neuritic_plaque(t)
```

As illustrated, object relational queries are more concise compared with their relational counterparts. Note that a major assumption that makes our comparison feasible is that predicate values can be pre-computed to improve performance. This assumption will not be valid if the number of combinations is excessive (e.g., similar patients over different time intervals). In that case predicate values have to be computed at run time and we
cannot find equivalent queries in the relational framework given an object relational query.

V. CONCLUSIONS

Neuroscience provides a complicated and interesting domain to stimulate the development of sophisticated object modeling techniques, database management, and query processing strategies. This paper summarizes our effort in applying the object relational database technology to facilitate intelligent data retrieval across different levels of abstraction in neurosciences. It demonstrates the advantages of employing the object relational database technology for a typical biomedical application, in terms of expressive power and performance.

We have also described a set of terminology that can be used to pose intelligent queries. The set of terminology is, of course, not standard and subject to change and expansions. The techniques described in [1] [3] may be applied to allow different, yet similar, terms to be employed.

REFERENCES


