Model Formulation ■

Automated Database Mediation Using Ontological Metadata Mappings

Luis Marenco, MD, Rixin Wang, PhD, Prakash Nadkarni, MD

Abstract **Objective:** To devise an automated approach for integrating federated database information using database ontologies constructed from their extended metadata.

Background: One challenge of database federation is that the granularity of representation of equivalent data varies across systems. Dealing effectively with this problem is analogous to dealing with precoordinated vs. postcoordinated concepts in biomedical ontologies.

Model Description: The authors describe an approach based on ontological metadata mapping rules defined with elements of a global vocabulary, which allows a query specified at one granularity level to fetch data, where possible, from databases within the federation that use different granularities. This is implemented in OntoMediator, a newly developed production component of our previously described Query Integrator System. OntoMediator's operation is illustrated with a query that accesses three geographically separate, interoperating databases. An example based on SNOMED also illustrates the applicability of high-level rules to support the enforcement of constraints that can prevent inappropriate curator or power-user actions.

Summary: A rule-based framework simplifies the design and maintenance of systems where categories of data must be mapped to each other, for the purpose of either cross-database query or for curation of the contents of compositional controlled vocabularies.

■ **J Am Med Inform Assoc.** 2009;16:723–737. DOI 10.1197/jamia.M3031.

Introduction

One challenge in federated database integration is that databases from various research groups may store information on the same category of data differently. *Physical* heterogeneity issues, such as different internal names for semantically equivalent tables/columns, data stored in a single table vs. multiple tables, data stored in column-modeled vs. row-modeled form, and so on, have been addressed successfully through standard approaches such as database views and mappings of individual database schema elements to a global schema. However, semantic representation differences—notably due to equivalent information being stored at different granularity—cannot be addressed using these mechanisms. For example, in one database, various attributes of a concept—e.g., morphology, location, tissue type—may be represented as distinct fields, while in another database these attributes may be combined implicitly through the descriptive name of that concept. This paper describes a general approach to specifying equivalence between different concepts when such granularity differences exist. We describe an implementation for integration of neuroscience databases, and provide another example in the biomedical controlled vocabulary domain. This work may lay the foundation for database-contextual information integration in biosciences and other areas of research.

Affiliations of the authors: Center for Medical Informatics (LM, RW), Department of Anesthesiology (LM), Yale University School of Medicine, New Haven, CT; Geisinger Health Systems (PN), Danville, PA.

This research is supported by NIH Grants R01 DA021253 and P01 DC04732.

The authors thank the curators of the CCDB and CoCoDat databases for making their data available for use in this paper, and Dr. Gordon Shepherd for curating the neuron ontology used in this work.

Correspondence: Luis Marenco, MD, Center for Medical Informatics, Yale University School of Medicine, PO Box 208009, New Haven, CT 06520-8009; e-mail: <luis.marenco@yale.edu>.

Received for review: 10/13/08; accepted for publication: 06/07/09.

Background

Approaches to Database Integration

Two general approaches to database integration are the warehouse approach, which emphasizes data translation, and the database federation approach, which emphasizes query translation.[1] In the warehouse approach, individual sources' data are converted to a common grain and moved to a read-only warehouse whose data model is the union of the source models. Query optimization, performance, and a priori data validation and curation are readily addressed, but this approach succeeds only with a high degree of central control, which is not seen in most collaborative research scenarios. It is also much less workable for research databases where schemas and metadata change constantly. In the federated database approach, only information about the individual sources' data models (schema metadata) is integrated centrally into a federated schema;[2,3] the data sources remain separate. "Mediator" software translates queries against the federated schema into queries against individual sources and merges the results. Differences in granularity, encoding, and representation semantics make

this approach challenging, in addition to limiting the quality and value of the output. While simple Web-browsing interfaces that show details of a single item of interest are feasible, representing sophisticated analytic queries involving complex Boolean logic that return numerous rows of data are rarely possible.

The Problem: Querying across Datasets of Heterogeneous Granularity

Successful query of a federated schema depends on being able to correctly identify—that is, "map"—equivalent concepts across different databases: varying data granularity across databases complicates the query formulation process. Fortunately, granularity variation is not entirely ad hoc: there exists a definite logical model underlying the representation of all of a given database's concepts. However, because this model is very often *implicit*, solving the mapping problem across databases requires creating an equivalent *explicit* representation.

Certain types of granularity heterogeneity are seen in databases that need to be integrated before metaanalysis.[4] Here, one encounters parameters of the *nominal* (enumerated) or *ordinal* (ranked) data type, whose valid values belong to a discrete set, but where the set members are defined differently for the same parameter in different databases. In this domain, it is well known that one can generally map a finer grain to a coarser grain but not vice versa. For example, if one dataset records "smoking status" as "Smoker/Non-Smoker", and another as "cigarettes/day", one can infer that anyone who smokes more than zero cigarettes per day is a smoker, but inferences in the reverse direction are not possible. Grain translations are achieved here through the straightforward approach of translation (lookup) tables that map the levels in one measure to levels in the other. While equivalent to if-then rules, table-driven methods have the advantage of being more efficient and easier to modify.[5]

The granularity problem described in this paper, which we believe has not been addressed previously in a systematic fashion, concerns *explicit* versus *implicit representation of concept attributes*. That is, attributes are represented explicitly as separate fields for a concept category in one database, but merged into the textual descriptions of those concepts in another database. We show that this problem is intimately linked to the issue of compositional versus noncompositional concept representation in controlled vocabularies, described below.

The Concept Representation Problem in the Biomedical Vocabulary Domain

To provide background for our approach for handling heterogeneity in concept attribute representation, we summarize a well-known issue in the controlled vocabulary field. There are two ways to combine new concepts from existing concepts as knowledge within a domain evolves. With *precoordination*, the vocabulary's *curators* create a new concept entry with a descriptive phrase that captures the meaning of a combination—e.g., "renal hypertension". With *postcoordination* (composition), the vocabulary's *power-users* combine existing concepts (here, "secondary hypertension" and "kidney disease") into a miniature semantic network using relationship edges (such as "Cause-of") from a set of permissible relationship types. The pros and cons of each approach are discussed by White et al.[6] With precoordination, while curatorial fiat weeds out nonsensical combinations, concepts can still proliferate. Tasks such as concept-based document searching and manual/electronic concept matching of text become much more complicated: highly complex concepts are very difficult to match exactly. A compositional vocabulary's contents are much less likely to proliferate, but a naive user may combine concepts in meaningless ways. To mitigate this risk, it is desirable to specify *computable* constraining rules for concept composition. However, to the best of our knowledge, no constraint-based computational framework is operational in the biomedical ontology domain.

- In the precoordinated vocabulary LOINC (logical observations, identifiers, names and codes),[7] a concept is defined based on a *fixed* combination of the parameter being recorded/measured, the property recorded, temporal aspects of the recording, the system in which the parameter was recorded (e.g., blood, urine), the recording/measuring method and the scale (data type) in which the result is expressed. That is, the compositional rule is *implicit* and *hard-coded* in the structure of LOINC's schema.
- With the compositional systematic nomenclature of medicine (SNOMED),[8] the composition of individual complex concepts in terms of others is recorded through pairwise relationships between a complex concept and one or more simpler concepts. Sometimes, a set of relationships is grouped together using an integer Relation Group field to indicate that they collectively form a single semantic unit. However, constraining rules that apply to entire *families* of concepts, such as the categories of data that a given attribute applies to, and the values allowed for that attribute, are defined *only in prose form* in the SNOMED user Guide.
- Finally, the web ontology language (OWL) is intended to support ontology development and interchange using an XML-based syntax. While OWL supports definition of certain constraints based on description logics,[9] the constraint syntax, which is fixed and nonextensible, does not support compositional rules to deal with concept-granularity heterogeneity.

Existing Approaches to Query Mediation

From the very extensive and diverse literature on query mediation, we summarize briefly previous approaches that either use metadata or ontologies to drive the query process, or those that focus on bioscience-related problems.

Several projects have explored database interoperation/integration issues in genomics.[10,11] BioMediator,[12,13] developed at the University of Washington, uses a federated approach to access a number of public genomics-related data sources, e.g., NCBI entrez,[14] Online Mendelian Inheritance in Man (OMIM),[15] and Gene Ontology.[16] The global data model treats both concepts and data as nodes in a semantic net. This work has subsequently been extended to the neuroscience domain.[17] In neuroscience, the UCSD BIRN (Biomedical Informatics Research Network) Database Mediator[18,19] uses a centralized data repository with remote views that collectively act as a global ontology. CaGrid[20,21]

uses Web-services and grid-computing technologies to support resource discovery and integrated data analysis for NCI's Cancer Biomedical informatics Grid (CaBIG) infrastructure.

Won Kim's classic 1991 paper[22] provided a taxonomy of representational discrepancies between databases. Cheung et al.[23] used these principles to map between two laboratory genomic databases and a public genomics database. Philip Bernstein's group at Microsoft Research has explored various algorithms for automated schema matching between different databases, comparing versions of the same database,[24] and facilitation of interactive matching by a curator.[25] Bernstein characterizes schema matching as a subproblem within the larger field of model management.[26] Semantic Web technologies have also been used to integrate scientific data. One approach[27] represents information in RDF and OWL[28] formats and stores it using Oracle Semantic Technologies, to be processed later by ontological reasoners. The SEMEDA[29] uses a federated database approach with relational database technology to annotate table/column descriptions with entries in a custom taxonomy, thereby allowing semantic integration based on columns across different databases with common semantics (e.g., EC numbers for enzymes).

None of these approaches attempt to address the problem of heterogeneous granularity.

Ontologies: A Working Definition

We describe our approach as based on ontological mappings. Because of wide variations in the usage of the term "ontology", which broadly refers to a database application that supports controlled terminologies, we clarify our use of the term by referring to the Wikipedia entry.[30] An ontology is a structured repository of:

- Concepts, and terms (synonymous forms) for those concepts. Some terms are designated as *preferred* terms for a given concept.
- Relationships, where a pair of concepts is connected by a "relationship-type", itself a special kind of concept. Some relationship types are semantic inverses of each other, e.g., "Part-of" and "Contains". The subset of relationships with "Is-a" links form a *concept hierarchy*.
- Classes, which categorize the information in the source systems, and properties (attributes) of each class, are roughly analogous to database tables and columns. Classes are typically organized into a class hierarchy, where subordinate classes are linked to superclasses through "Is-a" relationships.
- Optionally, computational assertions or rules support tasks such as concept classification.

Model Description

The motivation for the model was the need to provide integrated access to data modeled at different granularity across data sources, while at the same time allowing access to individual sources' data *at the grain of the individual models* when needed. In a data-federation scenario where, for example, three of four data sources record a particular class of data at a fine grain, while a fourth source uses a coarse grain, a global data model that discards the finer grain in the interests of overall homogeneity is unlikely to meet its users' needs. For a query expressed at a coarse grain, a useful system should return data from *all* data sources converted to that grain. For a query expressed at the fine grain, however, the system should return data from the three sources that support this grain.

- Our model is based on mappings that describe the transformation of a family of concepts in one database into semantically equivalent concepts in another, which are defined in a generic fashion using compositional-rule syntax.
- To support a federated query of heterogeneously represented data, such mappings are consulted by a query-generation framework at runtime, so the correct query operations are executed. Enabling the rule syntax to be *computable* requires a metadata subschema that records the data models of each database in the federation, as well as a generic controlled-vocabulary subschema that represents concepts, terms and relationships between concepts. Runtime efficiency of rule execution is facilitated by a compilation step that transforms the rule into stack-machine operations.
- The model also requires that concepts must be associated with concept categories or classes, and that the class definitions also record the description of the properties or attributes of each class.
- The model borrows ideas from the controlled-vocabulary domain, and when applied to compositional vocabularies, transformational mappings allow a computable representation of certain description-logic-based constraints that are specified noncomputably.

Background for Neuroscience Example: OntoMediator and The Query Integrator System

We have designed an implementation of the model (OntoMediator) in the neuroscience domain. To describe its operation adequately, we first describe the infrastructure of which it is a part. OntoMediator relies on our previously described Query Integrator System[31] (QIS) http://ycmi.med.yale.edu/QIS, summarized here. The QIS uses a distributed architecture composed of three intercommunicating categories of loosely coupled functional units. **Data source servers (DSS)**—gateways to individual research groups' data sources—contain schema-related metadata about the sources, including textual annotations and mappings to standard vocabularies. We use "data source" as a synonym for "a database's publicly accessible sub-schema". **Integrator servers (IS)** combine this metadata into a federated schema repository, support interactive query composition and also host stored queries, which can run interactively as well as in automated workflows through a Web-service mechanism. The **ontology servers (OS)** are now described in greater depth: see Fig 1.

An OS maintains the contents of one or more ontologies and their mappings to the metadata in DSS's databases to be used within the federation. Some of these ontologies are standard (reference) ontologies, e.g., the NeuroNames[32] subset of the unified medical language system (UMLS) metathesaurus,[33] while others are *data-source-specific* and maintained by the individual research groups. The ontologies also contain database-related mapping information. Thus, certain concepts map to Classes in certain databases, while other concepts map to attributes, or class instances ("objects"), which are roughly analogous to rows in a table.

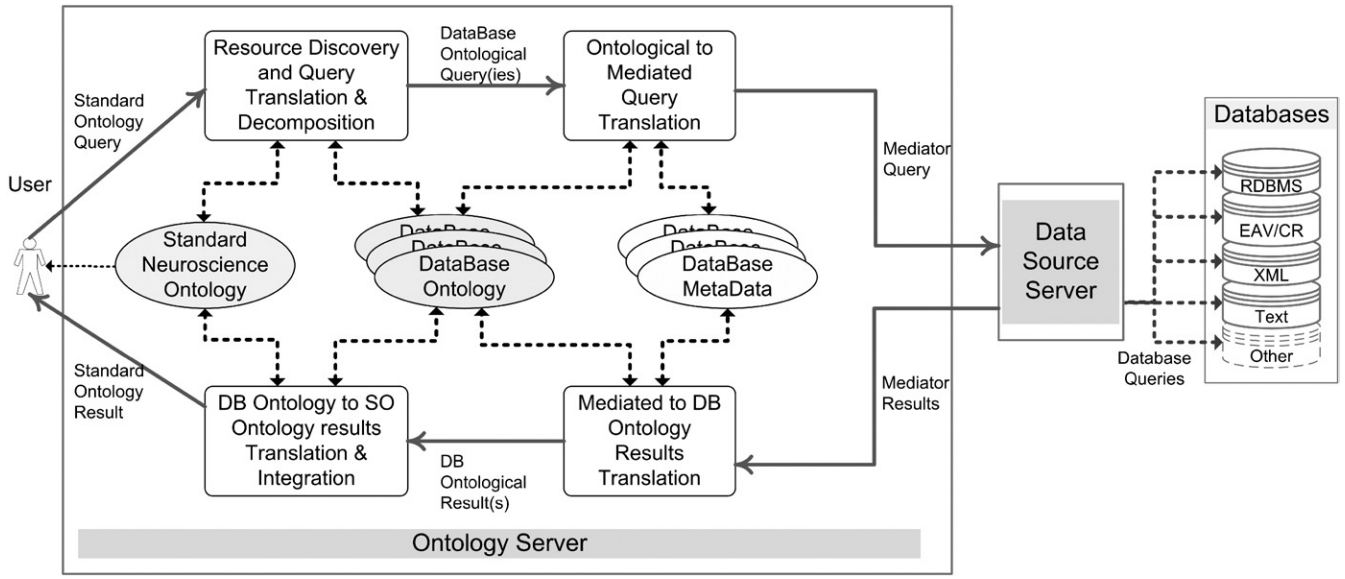

Figure 1. Processes and components involved in the ontological federated database mediation: The user selects and composes a query using a standard ontology as federated schema, this query is then decomposed and translated into database specific ontological and then language specific subqueries. These subqueries are executed against the databases themselves and their results are translated back into terminology and ontology structure used as federated schema.

For example, the concept "Neuron" or "Ion channel" may map to specific classes, whereas "Cerebellar Purkinje neuron" refers to a Neuron object. The mapping information allows the OS to act as an information source map (ISM).[34] Parts of the database schemas of the DSS, IS, and OS systems are replicated for efficiency reasons, to minimize unnecessary communication over the Internet; and to synchronize database schema changes. All the QIS servers communicate with each other via Web services.

A user querying the system can designate any ontology within the OS as the "default" ontology (i.e., *the default source of preferred terms*). The default ontology will typically be the data-source-specific ontology that the user is most familiar with, but may also be a reference ontology. Choosing a default ontology has the following benefits:

1. It supports *interactive query formulation*: the query interface can configure itself so the choices presented to the user for selection of data classes and attributes will use the phrases and definitions that are preferred by the user.
2. *Query composition* in terms of the underlying database language is facilitated by the mappings of terms in the default ontology to concepts in the reference ontology, and thence to the individual ontologies and database mappings of the individual data sources.
3. It facilitates *integrated data display*: the results returned after query execution (e.g., column headings, row values) are presented using the default ontology's terms.

Features 1 and 3 are potentially useful in international collaboration scenarios, where the user interface can self-configure to use the terms of the user's preferred language.

OntoMediator's query formulation user interface is Web based and employs a hierarchical-query-by-example metaphor. "Hierarchical" means that the user can drill down the federated schema's class hierarchy, to the level of individual selectable attributes. The example query accesses three production neuroscience data sources:

- NeuronDB, part of Yale's SenseLab project,[35,36] contains data and supporting literature about neuronal membrane properties (receptors, transmitters, and currents/channels) localized to specific neuronal compartments of different types of neuron.
- CCDB (Cell Centered Database), developed at the University of California at San Diego (UCSD),[37,38] contains microscopy and microscopy-related data from several types of nervous system cells.
- CoCoDat (Collation of Cortical Data), built at the C. &. O. Vogt Brain Research Institute in Düsseldorf, Germany[39] contains experimental data, including membrane properties, electrophysiological data, neuron connectivity, and supporting literature focused on specific subareas of the cerebral cortex.

These databases are accessible via a DSS located at http://dss-qis.med.yale.edu. This link can be used to browse the database imported metadata, terminological data and metadata annotations. At the OS, each database is associated with a database ontology. These ontologies were bootstrapped from the associated database/schema metadata and then augmented manually with annotations by us. The ontology server itself uses http://Microsoft.net 2.0 framework, Internet Information Services and a SQL-Server 2008 database.

The default ontology used for the example is Neuroscience-So, a reference ontology developed by us in collaboration with Yale neuroscience researchers. It can be browsed via the URL http://os-qis.med.yale.edu/ontoDashboard/?sid=12. The subset of Neuroscience-SO's ontology related to neurons includes 33 selected types of neuron, hierarchically organized. Neuronal compartments include the axon, soma, as well as dendritic compartments (e.g., proximal, medial, and distal) for different types of dendrite (e.g., apical and basal). Neuronal properties are a superset of those contained in the three databases.

The neuron-related records/objects in NeuronDB are analogous to precoordinated concepts, while the objects in CoCoDat and CCDB are analogous to postcoordinated concepts in that they are based on a combination of properties such as cell type, region and sub-region. To map records in one database to equivalent records in another, we implement mappings at two levels.

- We first define broad mapping/compositional *rule definitions* ("RuleDefs") that define families of concepts (i.e., *classes*) in terms of a combination of other concept families.
- We then define actual instances of these definitions ("rules") in terms of specific concepts and specific property values (which are themselves concepts of a different kind).

The rules themselves are analogous to SNOMEDs approach of defining precoordinated concepts in terms of other concepts. The high-level rule definition acts as a template whose placeholders are instantiated in individual rules using actual concept IDs: rule definitions have no analogue in SNOMED. Multiple rules can use the same rule definition as template. The rule definitions are expressed using a yacc grammar,[40] described shortly.

In terms of the implementation, RuleDefs are used to map database schema elements (classes/tables and attribute/columns), i.e., concepts based on schema *metadata*, while the *rules* based on a given RuleDef map actual field values, which are concepts based on *data*. (Note that our usage of "rule" is different from that in expert systems, where if-then rules can form the basis for automated inferencing. It may be less confusing to think of them as "cross-database object equivalences that utilize patterns specified in terms of class structure".)

Concepts Mappings Grammar

Both Ruledef and Rules use the same syntax with some limitations for the RuleDefs. Ruledefs are expressed in terms of Classes. Rules are expressed in terms of Objects (instances of Classes).

RuleDef Grammar:

Class → ClassReference ': ' ConceptID

Class → Class ClassList

/* "ClassReference" is a sequential number used to correlate Classes and class instances between Ruledef and Rules. */

Class list → NULL/* the empty string */

| ListItem

| ClassList ListItem

| '[' ClassList ']'

ListItem → operator Class

operator → RelationshipType | '+'

Explanation. A class is defined as a combination of one or more class-references, separated by operators. Sub-expressions after the first class-reference may be enclosed in square brackets to indicate that the subexpression is optional (i.e., can occur zero or one times), and brackets may be nested.

There are two kinds of operators:

- *RelationshipType*, as in SNOMED, is a concept of type *relationship-type*, e.g., "part-of", "associated-with". A relationship allows us to include information that relates classes of concepts to each other in a way that is meaningful in terms of the knowledge domain.
- "+" (plus) is the composition operator. This is typically used for database mappings that use column/property information, where the RuleDef developer has determined that no special semantics need be applied to the combination of classes.

Purpose and Example. A Ruledef is a compositional rule template that provides a broad mapping of a class in one database to a class or classes in another. For example, the Ruledef

Neuron → Neuron_type [+ Anatomic_region [+ Layer_in_region]]

Maps a Neuron class (in NeuronDB) to the Neuron Type, Anatomic Region and Layer classes in CoCoDat and CCDB. The anatomic region and the layer are optional: some neuron-types are specific to some part of the nervous system, and layers may apply to specific areas of the nervous system but not to others.

Rule Grammar:

ClassInstance → ClassConceptExpr [operator ClassConceptExpr]

ClassConceptExpr → ClassReference ': ' ConceptExpression

ConceptExpression → ConceptID

| ConceptID '^'

| '('ConceptExpression','ConceptID')'

Summary

A *class instance* is one or more *class-concept expressions* separated by operators. A *class-concept expression* consists of a *class reference* and *concept expression* separated by a colon. A *concept expression* is either a ConceptID that corresponds to the *name* of a concept, optionally followed by a caret, or a parenthesized, comma-separated list of such ConceptIDs. Basically, the structure of the Rule is dependent of the structure defined by the RuleDef from which it is derived.

ClassReference and *operator* have the same definition as in the RuleDef.

We now provide an explanation and examples of the rule grammar. The symbols are described below:

1. ': ' (colon)—indicates *membership* of an instance in a class, e.g., *Neuron-type*: *Pyramidal* Neuron refers to the "Pyramidal Neuron" object that is a member of the class "Neuron_type".
2. '^' (caret) –the "And descendants" operator indicates that that the rule applies to a concept as well as to the hierarchical descendants of that concept in the concept hierarchy. The ability to specify descendants prevents proliferation of multiple rules with near-identical semantics.
3. ','(comma)—the *List* operator is used to combine a list of concepts, indicating that the rule applies to any one of them.

Purpose and Example

A RuleDef is a broad constraining/mapping mechanism, but it is not intended to tell us which individual instances of the Neuron class are mapped to instances in other databases.

The *rules* based on this rule def provide this information. For example,

Neuron: Deep Neocortical Pyramidal Neuron →

Neuron_type: Pyramidal Neuron + *Brain_region*: General Cortex^ + (Layer 5, Layer 6)

This rule states that the Deep Pyramidal Neuron Object (in the Neuron Class) corresponds to the database record in CocoDat, where the Neuron Type property has the value Pyramidal Neuron, the Brain Region property has the value "General Cortex" (or any concept that is a hierarchical descendant of General Cortex), and the Layer-in-Region property has either of the values "Layer 5" or "Layer 6".

Internal Implementation. Internally, for efficiency purposes, RuleDefs and rules are stored in a relational table after being translated algorithmically by an infix-to-postfix expression converter into a stack-machine equivalent of the RuleDef/rule expression, in much the same way that modern mathematical calculators convert parenthesized arithmetic/algebraic expressions before evaluation. The references to classes and concepts refer to numeric IDs in the ontology data/metadata: classes are treated as a special category of concept. However, in the user interface for rule/RuleDef creation, the developer (who is a domain expert) works with the equivalent descriptive phrases/concept names, which are selected through a search interface or pull-down lists.

Strictly speaking, the class references in a rule are redundant, because once a RuleDef template is specified, the appropriate class reference for a given class instance can be determined based on its position in the rule. However, a characteristic of ontologies is that the same phrase may refer to multiple concepts (e.g., "5HT" may refer to the neurotransmitter molecule or to the family of receptors for this molecule). The class reference prefix makes the human interpretation of the rule clearer because it makes it obvious which particular concept we are referring to. In terms of the rule creation interface, the presence of a given class reference constrains the values of the corresponding concept-instances that are presented in a pick-list.

These rules can be explored online at the following URL: http://os-qis.med.yale.edu/ontoDashboard/mappingRuleEditor.aspx.

Internally, the RuleDef and Rules information is stored in relational tables. This allows indexing of the RuleDef IDs and Concept IDs, and logarithmic lookup versus linear lookup if they were stored as text. (This approach is similar to SNOMEDs approach for storing the information about composition of a given complex concept in multiple rows in the Relationships table.)

Application of the Model: OntoMediator Query

We illustrate the model's application with a query that bridges the three federated databases mentioned earlier: it retrieves information about the physiological properties and morphology of dendrites. The query is "find dendritic-spine information and images and/or gK2 peak conductance values for all "neurons" having GABA-A receptors present at any dendrite compartment". To show this query's operation, we show the relevant subset of the conceptual federated schema in the Entity-Relationship diagram of Fig 2a. This figure's five tables figure exist in three separate physical databases: one table (Neuron) has physical equivalents in all three. In terms of the federated data model, the query is:

SELECT Neuronal-Compartment. Name, Receptor. Name, Ionic-Conductance. Name, Ionic-Conductance. Peak-Conductance, Neuronal-Structure. Name, Neuronal-Structure. Image.

From Receptor INNER JOIN (Neuronal-Structure right JOIN ((Ionic-Conductance right JOIN Neuronal-Compartment ON Ionic-Conductance. Conductance-ID = Neuronal-Compartment. Ionic-Conductance) Inner JOIN Neuron ON Neuronal-Compartment. Compartment-ID = Neuron. Compartment) ON Neuronal-Structure. Structure-ID = Neuron. Structure) ON Receptor. Receptor-ID = Neuronal-Compartment. Input-Receptor.

Where Neuronal-Compartment. Name like "% dendrite%" And receptor. Name = "GABA-A Receptor" And Ionic-Conductance. Name = "gk2" And neuronal-Structure. Name = "dendritic spine";

The query involves two outer joins. Detail information on Ionic Conductance in CocoDat and on Neuronal Structure in CCDB is not guaranteed to be present for all neurons present

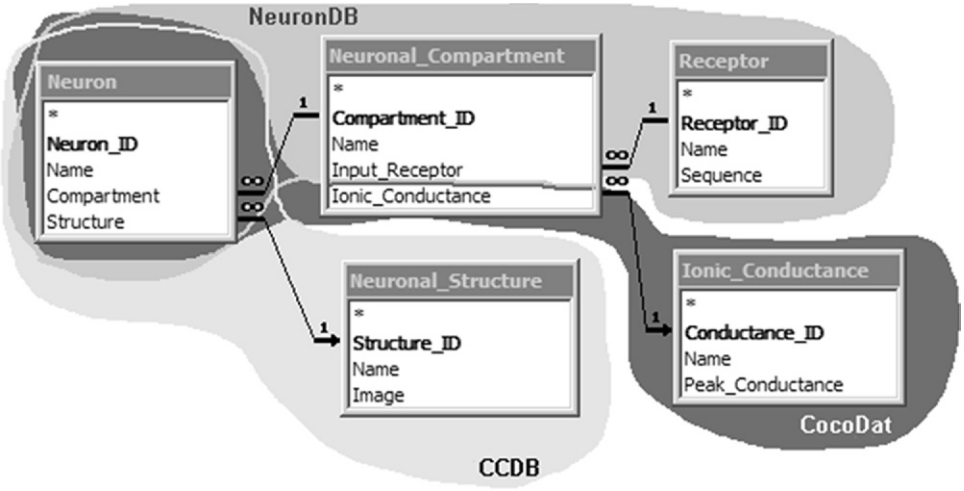

Figure 2a. An entity-relationship representation of the conceptual "schema" that is accessed by the sample federated query. The tables enclosed in the dark, medium and light backgrounds colors are physically located in the databases NeuronDB, CoCoDat and CCDB respectively. The table Neuron, where the three backgrounds overlap, is a conceptual table with physical equivalents in all three databases.

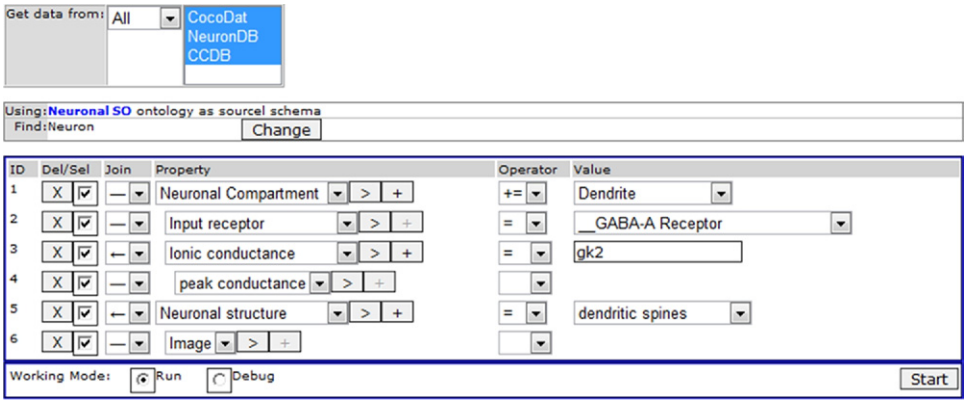

Figure 2b. The query as composed in the OntoMediator interface. At the top are selected the databases queried, next the federated schema used for querying the information "Neuronal SO", then, besides the find label, the entity being searched: "Neuron". At the bottom, a table showing the list of parameters joins and conditions used in the query. The user specifies the Class of interest by entering "% neuron%" in the "Find" text box ("%" is the SQL wildcard symbol), clicking "Go" and selecting "neuron" from the list of classes retrieved.

1. OntoMediator fills-in the first parameter row with default values. The user now modifies the parameters to formulate the desired query.
   a. The "Del" button lets you delete a row if accidentally specified.
   b. The "Select" checkbox indicates that the property/column is to be displayed.
   c. The join pull-down lets you specify either the default inner join (–) or an outer join (←).
   d. The "Property" pull-down's contents are limited on the current class being operated upon.
   e. To add another row to the query parameters, the user can click either the ">" button or the "+" button. ">" lets you access another property in the same class. The "+" button is active only when the chosen property itself is a reference to a class, and you want to specify a sub-property. (this is equivalent to navigating to a related table and accessing its columns in turn.)
   f. The list of operators is "−", ">", "<" or "like" for string properties.
   g. The value field is usually a text box; it morphs into a pull-down when the property is limited to a set of discrete values or is a class reference. In the latter case, the pull-down shows names of all instances (objects) for that class. In the example query, the items, "dendrite", "gaba-A receptor" and "dendritic spine" were selected from pull-downs.
2. After query composition, the user clicks start. Debug mode option permits single-stepping through the query.

in the federated schema, and so we include neurons that may lack this information. The reader can compose/execute this query c via http://os-qis.med.yale.edu/QMap/?sid=12. This link already specifies Neuroscience-SO as the federated ontology to use for the query. Upon loading, OntoMediator's screen shows a list of databases to be searched, by default all are selected. Fig 2b shows the query composition interface. The processes involved during query execution are now described:

1. **Federated ontological query composition:** Figure 3 shows the federated query automatically generated by OntoMediator, which implements a query language resembling SQL, but with an XML flavor in its use of attribute-value pairs, which makes it easier to parse. The attributes "tid" and "vid" refer to "term id" and "value id" in the ontology being used. (Term ID is a standard component of large-scale ontologies: Value ID applies to IDs of class instances in the data.)
2. **Query decomposition:** The federated query is decomposed into several data-source-specific queries, by checking which classes belong to each data source and which are common. The resulting queries are shown in Fig 4a. The data source for each query is identified in the "from" section of the query.

```
ontology_query id= "q1" type= "SO Query"
  using id= "s12" name= "Neuronal SO" type= "Standard"
  select
    set id= "r0" tid= "t1225" tname= "Neuron" show= "true"
      set id= "r1" tid= "t1067" join= "inner" tname= "Neuronal Compartment" condition= "+=" value= "Dendrite" vid= "t1075" show= "True"
      set id= "r2" tid= "t1012" join= "inner" tname= "Input receptor" condition= "=" value= "__GABA-A Receptor" vid= "t1229" show= "True"
      set id= "r3" tid= "t1333" join= "left" tname= "Ionic conductance" condition= "=" value= "gk2" show= "True"
        set id= "r4" tid= "t7934" join= "inner" tname= "peak conductance" show= "True"
    set id= "r5" tid= "t1462" join= "left" tname= "Neuronal structure" condition= "=" value= "dendritic spines" vid= "t1463" show= "True"
      set id= "r6" tid= "t1467" join= "inner" tname= "Image" show= "True"
  from
    source id= "s5" name= "CocoDat" type= "DSS"
    source id= "s9" name= "NeuronDB" type= "DSS"
    source id= "s24" name= "CCDB" type= "DSS"
```

Figure 3. The ontological query represented in the internal QIS-XML syntax. Most of the attributes are self-explanatory when comparing them with the visual interface in Fig 2b. With the exeption of "tid" and "Vid" used for term id and value id The preferred ids for tname and value.

```
querySet
  ontology_query id= "q1_1" type= "DB-SO Subquery"
    using id= "s12" name= "Neuronal SO" type= "Standard"
    select
      set id= "r0" tid= "t1225" tname= "Neuron" show= "true" ruledef_id= "1"
        set id= "r1" tid= "t1067" join= "inner" tname= "Neuronal Compartment" condition= "+=" value= "Dendrite" vid= "t1075" show= "True"
          set id= "r3" tid= "t1333" join= "left" tname= "Ionic conductance" condition= "=" value= "gk2" show= "True"
            set id= "r4" tid= "t7934" join= "inner" tname= "peak conductance" show= "True"
    from
      source id= "s5" name= "CocoDat" type= "DSS"
  ontology_query id= "q1_2" type= "DB-SO Subquery"
    using id= "s12" name= "Neuronal SO" type= "Standard"
    select
      set id= "r0" tid= "t1225" tname= "Neuron" show= "true"
        set id= "r1" tid= "t1067" join= "inner" tname= "Neuronal Compartment" condition= "+=" value= "Dendrite" vid= "t1075" show= "True"
          set id= "r2" tid= "t1012" join= "inner" tname= "Input receptor" condition= "=" value= "__GABA-A Receptor" vid= "t1229" show= "True"
    from
      source id= "s9" name= "NeuronDB" type= "DSS"
  ontology_query id= "q1_3" type= "DB-SO Subquery"
    using id= "s12" name= "Neuronal SO" type= "Standard"
    select
      set id= "r0" tid= "t1225" tname= "Neuron" show= "true" ruledef_id= "1"
        set id= "r5" tid= "t1462" join= "left" tname= "Neuronal structure" condition= "=" value= "dendritic spines" vid= "t1463" show= "True"
          set id= "r6" tid= "t1467" join= "inner" tname= "Image" show= "True"
    from
      source id= "s24" name= "CCDB" type= "DSS"
```

Figure 4a. The federated query gets decomposed into federated subqueries targeting each database. Elements of the federated query not found in the target database are being removed.

3. **Term/object ID/structural transformation:** In each resultant query, for Classes where an exact correspondence exists between federated and data-source schemas, the term and value/object IDs are transformed into their corresponding source database via the Ontology's global concepts table. For terms lacking this correspondence; we use the RuleDefs and Rules for these terms to identify the corresponding data-source terms (i.e., we retrieve the right hand-side entries corresponding to a left-hand-side entry). Figure 4b shows the transformed database ontological queries. In addition to changes in term ids/names, note how the set for "neuron" in the federated query was transformed into a series of node-sets for CocoDat (Neuron Type, Brain Region and Region Layer) and CCDB (Nervous system cell and Region in Organ).
4. **Translation of conceptual OntoMediator queries into QIS queries by the OS, using individual schema's data models:** In some cases (not covered in this paper), one conceptual OntoMediator query may spawn more than one database query. In the present example, CocoDat and CCDB use a traditional relational data model, hence their query composition plan involves the following steps:
   - Fetch all tables involved in the query.
   - Select the sets from which data has been requested.
   - Join the table using their imported or annotated relationships, using intermediate tables/relationships in the schema to link tables that are not directly related.
   - If all values in a hierarchy are required, expand the values to include its descendants. Note how "Dendrite" is expanded in the CocoDat query to include "Dendrite | Apical Dendrite | Basal Dendrite | Distal dendrite | Proximal dendrite".

```
querySet
  ontology_query id= "q1_1_1" type= "DBO Query" idref= "q1_1"
    using id= "s5" name= "CocoDat" type= "DSS"
    select
      set id= "r1" tid= "t790" tname= "NeuronType" show= "true" ruledef_id= "1" alias= "r0.0"
        set id= "r2" tid= "t786" tname= "BrainRegion" show= "true" alias= "r0.1"
        set id= "r3" tid= "t882" tname= "RegionLayer" show= "true" alias= "r0.2"
        set id= "r4" tid= "t789" join= "inner" tname= "Compartment" condition= "+=" value= "Dendrite" vid= "t912" show= "True" alias= "r1"
          set id= "r5" tid= "t834" join= "left" tname= "Ionic conductance" condition= "=" value= "gk2" show= "True" alias= "r3"
            set id= "r6" tid= "t837" join= "inner" tname= "Peak_conductance" show= "True" alias= "r4"
  ontology_query id= "q1_2_1" type= "DBO Query" idref= "q1_2"
    using id= "s9" name= "NeuronDB" type= "DSS"
    select
      set id= "r1" tid= "t1" tname= "Neuron" show= "true" alias= "r0"
        set id= "r2" tid= "t7" join= "inner" tname= "CF_Compartment" condition= "+=" value= "Dendrite" vid= "t1286" show= "True" alias= "r1"
          set id= "r3" tid= "t8" join= "inner" tname= "Receptor" condition= "=" value= "GabaA" vid= "t183" show= "True" alias= "r2"
  ontology_query id= "q1_3_1" type= "DBO Query" idref= "q1_3"
    using id= "s24" name= "CCDB" type= "DSS"
    select
      set id= "r1" tid= "t7834" tname= "Nervous system cell" show= "true" ruledef_id= "1" alias= "r0.0"
        set id= "r2" tid= "t7863" tname= "Region in organ" show= "true" alias= "r0.1"
        set id= "r3" tid= "t7869" join= "left" tname= "Cellular structure" condition= "=" value= "spiny
dendrite" vid= "t7867" show= "True" alias= "r5"
          set id= "r4" tid= "t7899" join= "inner" tname= "Image base name" show= "True" alias= "r6"
```

Figure 4b. The federated subqueries are transformed into database ontological queries. For straight-mapped classes, terms are transformed into the source terms; for terms mapped using RuleDefs, the federated class is transformed using the rules syntax. See how the class neuron is transformed into several subclasses in CocoDat and CCDB.

SenseLab uses the EAV/CR (entity-attribute-value with classes and relationships) semantic data model,[41,42] so query composition involves the steps below:

- Select all classes involved in the query. If more than one class is needed, and is not directly related via an attribute, create one QIS query for each of those classes.
- Select each attribute to be retrieved.
- Create conditions, and expand hierarchical values if necessary (as above).

5. **QIS query execution**: Each QIS query is forwarded to the DSS harboring the connection to the data source. At each DSS, the QIS query is transformed into the underlying language of the source (in all three databases, flavors of SQL). Each query is executed asynchronously. Back on the OS the results are retrieved in XML and stored locally in a temporary space for use in the next transformation steps. Columns in these results are labeled with ids that, while cryptic to a user, allow subsequent ontological data integration.
6. **Database ontology results transformation** (Fig 5). Here, the column labels (and certain data values that are mapped to the ontology) are substituted with the corresponding phrases in the ontology used for the query. Figure 6 shows the partial results, using hierarchical tables in HTML.
7. **Federated ontological results transformation**: Database ontological-query results are transformed individually into federated-query results using one-to-one mappings and reverse-mapping Rules (i.e., determining the Left-hand-side entries given a combination of right hand-side values). Some examples of Rules transformation include converting "(Pyramidal Neuron, Temporal Lobe, Layer 2/3 Isocortex)" from CocoDat into "Neocortical Pyramidal Neuron: Superficial", and "Medium spiny neuron, neostriatum (CCDB)" into "Neostriatal spiny neuron". See Fig 7.

```
querySet
  QIS-Query ontology_query_id= "q1_1_1"
    info
      query id= "q1_1_1_1" name= "CocoDat" description= "Auto generated" owner= "OS" database= "CocoDat" serverdsID= "d56"
    from
      set id= "g1" gId= "QIS_NeuronTypes" name= "QIS_NeuronTypes" alias= "" version= "1" EID= "30377"
      set id= "g2" gId= "QIS_BrainRegions" name= "QIS_BrainRegions" alias= "" version= "1" EID= "30370"
      set id= "g3" gId= "QIS_RegionLayer" name= "QIS_RegionLayer" alias= "" version= "1" EID= "30391"
      set id= "g4" gId= "QIS_Compartments" name= "QIS_Compartments" alias= "" version= "1" EID= "30510"
      set id= "g5" gId= "Neurons_IonicConductances" name= "Neurons_IonicConductances" alias= "" version= "1" EID= "26353"
      set id= "g6" gId= "QIS_RecordingSites" name= "QIS_RecordingSites" version= "2" EID= "30456"
    select
      atom id= "c1" cId= "g1.NeuronType" name= "NeuronType" dt= "a200" alias= "r1" fromset= "g1"
      atom id= "c2" cId= "g2.BrainRegion" name= "BrainRegion" dt= "a200" alias= "r2" fromset= "g2"
      atom id= "c3" cId= "g3.RegionLayer" name= "RegionLayer" dt= "a200" alias= "r3" fromset= "g3"
      atom id= "c4" cId= "g4.Compartment" name= "Compartment" dt= "a200" alias= "r4" fromset= "g4"
      atom id= "c5" cId= "g5.Conductance_name" name= "Conductance_name" dt= "a200" alias= "r5" fromset= "g5"
      atom id= "c6" cId= "g5.Peak_conductance" name= "Peak_conductance" dt= "a200" alias= "r6" fromset= "g5"
    conditions
      cond id= "n1" cId= "g4.Compartment" name= "Compartment" dt= "a200" operator= "=" value= "Dendrite|Apical Dendrite|Basal Dendrite|Distal
dendrite|Proximal dendrite" calias= "r4" fromset= "g4"
      cond id= "n2" cId= "g5.Conductance_name" name= "Conductance_name" dt= "a200" operator= "=" value= "gk2" calias= "r5" fromset= "g5"
    join
      pair id= "j1" attA_Id= "g2.BrainRegion" attA_name= "BrainRegion" join_type= "inner join" attB_Id= "g6.BrainRegion" attB_name= "BrainRegion"
      pair id= "j2" attA_Id= "g3.RegionLayer" attA_name= "RegionLayer" join_type= "inner join" attB_Id= "g6.RegionLayer" attB_name= "RegionLayer"
      pair id= "j3" attA_Id= "g6.NeuronSiteID" attA_name= "NeuronSiteID" join_type= "inner join" attB_Id= "g5.ID_Neurons" attB_name= "ID_Neurons"
      pair id= "j4" attA_Id= "g4.Compartment" attA_name= "Compartment" join_type= "inner
join" attB_Id= "g6.NeuronCompartment" attB_name= "NeuronCompartment"
      pair id= "j5" attA_Id= "g1.NeuronType" attA_name= "NeuronType" join_type= "inner join" attB_Id= "g6.NeuronType" attB_name= "NeuronType"
    expression
  QIS-Query ontology_query_id= "q1_2_1" query_plan= "EAV/CR Rule #1"
    info
      query id= "q1_2_1_1" name= "NeuronDB" description= "Auto generated" owner= "OS" database= "NeuronDB" serverdsID= "d54"
    from
      set id= "g1" gId= "c25" name= "ncr" alias= "" version= "2" EID= "20923"
    select
      atom id= "c1" cId= "g1.a60" name= "a60" dt= "a200" alias= "r1"
      atom id= "c4" cId= "g1.a47" name= "a47" dt= "a200" alias= "r2"
      atom id= "c7" cId= "g1.a49" name= "a49" dt= "a200" alias= "r3"
    conditions
      cond id= "n1" cId= "g1.a47" name= "a47" dt= "a200" operator= "=" value= "Dendrite|Equivalent dendrite|Apical dendrite|Basal dendrite|Distal
basal dendrite|Middle basal dendrite|Proximal basal dendrite|Distal apical dendrite|Middle apical dendrite|Proximal apical dendrite|Distal equivalent
dendrite|Middle equivalent dendrite|Proximal equivalent dendrite"
      cond id= "n4" cId= "g1.a49" name= "a49" dt= "a200" operator= "=" value= "GabaA"
    join
    expression
  QIS-Query ontology_query_id= "q1_3_1"
    info
      query id= "q1_3_1_1" name= "CCDB" description= "Auto generated" owner= "OS" database= "CCDB" serverdsID= "d65"
    from
      set id= "g1" gId= "ANATOMICALDETAIL_OBJTAB" name= "Anatomical detail" alias= "" version= "1" EID= "94452"
      set id= "g2" gId= "MICROSCOPYPRODUCT_OBJTAB" name= "Microscopy product" alias= "" version= "1" EID= "140065"
    select
      atom id= "c1" cId= "g1.CELLTYPE" name= "CELLTYPE" dt= "a200" alias= "r1" fromset= "g1"
      atom id= "c2" cId= "g1.REGION" name= "REGION" dt= "a200" alias= "r2" fromset= "g1"
      atom id= "c3" cId= "g1.STRUCTURE" name= "STRUCTURE" dt= "a200" alias= "r3" fromset= "g1"
      atom id= "c4" cId= "g2.IMAGEBASENAME" name= "IMAGEBASENAME" dt= "a200" alias= "r4" fromset= "g2"
    conditions
      cond id= "n1" cId= "g1.STRUCTURE" name= "STRUCTURE" dt= "a200" operator= "=" value= "spiny dendrite" calias= "r3" fromset= "g1"
    join
      pair id= "j1" attA_Id= "g2.MICROSCOPYPRODUCT_ID" attA_name= "MICROSCOPYPRODUCT_ID" join_type= "inner
join" attB_Id= "g1.MICROSCOPYPRODUCT_ID" attB_name= "MICROSCOPYPRODUCT_ID"
    expression
```

Figure 5. The resulting QIS structured queries, after transforming the database ontological queries. These queries are sent to their specific DSS servers for one more transformation step into DB specific queries and their execution.

OntoQuery id:q1_1_1 Source Name:**CocoDat**

NeuronType

[-]	BrainRegion	RegionLayer	Compartment	
Pyramidal Neuron	Temporal lobe	Layer 2/3, Isocortex	[-]	Ionic conductance
			Distal dendrite	[-] Peak_conductance: gK2 0.1 mS/cm^2
			Proximal dendrite	[-] Peak_conductance: gK2 0.1 mS/cm^2

OntoQuery id:q1_2_1 Source Name:**NeuronDB**

Neuron

[-]	CF_Compartment	
Olfactory cortex pyramidal neuron	[-]	Receptor
	Distal apical dendrite	GabaA
	Proximal apical dendrite	GabaA
Olfactory bulb mitral cell	[-]	Receptor
	Distal apical dendrite	GabaA
	Middle apical dendrite	GabaA
	Proximal apical dendrite	GabaA
	Distal basal dendrite	GabaA
	Middle basal dendrite	GabaA
	Proximal basal dendrite	GabaA
Cerebellar purkinje cell	[-]	Receptor
	Distal equivalent dendrite	GabaA
	Middle equivalent dendrite	GabaA
	Proximal equivalent dendrite	GabaA
Neocortical pyramidal neuron: superficial	[-]	Receptor
	Distal apical dendrite	GabaA
Neocortical pyramidal neuron: deep	[-]	Receptor
	Proximal apical dendrite	GabaA
	Middle apical dendrite	GabaA
	Distal apical dendrite	GabaA
Retinal ganglion cell	[-]	Receptor
	Distal equivalent dendrite	GabaA
	Middle equivalent dendrite	GabaA
	Proximal equivalent dendrite	GabaA
Olfactory projection neuron	[-]	Receptor
	Middle equivalent dendrite	GabaA
Neostriatal cholinergic interneuron	[-]	Receptor
	Distal equivalent dendrite	GabaA

OntoQuery id:q1_3_1 Source Name:**CCDB**

Nervous system cell

[-]	Region in organ	Cellular structure	
medium spiny neuron	nucleus accumbens	[-]	Image base name
		spiny dendrite	ACC1_2ma
Purkinje neuron	cerebellum	[-]	Image base name
		spiny dendrite	pccor10_dc
medium spiny neuron	neostriatum	[-]	Image base name
		spiny dendrite	oka4 osaka4 wt_g8T6

Figure 6. Database ontological queries results, presented in hierarchical tables.

8. **Federated ontological results integration**: the result sets are combined by applying inner-join constraints to remove nonmatching rows. In Fig 8, two dendritic spine image-sets have been removed from the neostriatal spiny neuron and the medium spiny neuron: nucleus accumbens, for not having GABA-A receptors in their dendritic compartments. The peak conductances from CocoDat and images information from CCBD complement the constrained data from NeuronDB.

Query Performance: Benchmark

The example query takes 7.31 seconds to run. This duration comprises:

- Translation of the query into data-source-specific queries: 0.84 seconds. This phase includes execution of the compiled rules.
- Execution of individual queries and return of results to OS server: CocoDat 0.33 seconds, NeuronDB 2.34 seconds, and CCDB 3.2 seconds. Because of multithreading, individual queries' execution overlaps in time and the query execution phase takes 5.06 seconds. This includes time to reformat the data into XML and transfer the result to the OS, and includes a 1-second thread sleep period to verify completion of all queries.
- Merging of results/OS transformation using the default ontology: 1.41 seconds.

OntoQuery id:q1_1 Source Name:**Neuronal SO**

Specific neuron			
[-]	Neuronal Compartment		
Neocortical Pyramidal Neuron: Superficial	[-]	Ionic conductance	
	Distal dendrite	[-]	peak conductance
		gK2	0.1 mS/cm^2
	Proximal dendrite	[-]	peak conductance
		gK2	0.1 mS/cm^2

OntoQuery id:q1_2 Source Name:**Neuronal SO**

Specific neuron		
[-]	Neuronal Compartment	
Olfactory Cortex Pyramidal Neuron	[-]	Input receptor
	Dad	GabaA
	Dap	GabaA
Olfactory Bulb Mitral Cell	[-]	Input receptor
	Dad	GabaA
	Dam	GabaA
	Dap	GabaA
	Dbd	GabaA
	Dbm	GabaA
	Dbp	GabaA
Cerebellar Purkinje Cell	[-]	Input receptor
	Ded	GabaA
	Dem	GabaA
	Dep	GabaA
CA1 Pyramidal Neuron	[-]	Input receptor
	Dam	GabaA
	Dap	GabaA
	Dbd	GabaA
	Dbm	GabaA
	Dbp	GabaA
Dentate granule cell	[-]	Input receptor
	Ded	GabaA
	Dem	GabaA
	Dep	GabaA
Thalamic Relay Neuron	[-]	Input receptor
	Dem	GabaA
	Dep	GabaA
CA3 Pyramidal Neuron	[-]	Input receptor
	Dam	GabaA
Neocortical Pyramidal Neuron: Superficial	[-]	Input receptor
	Dad	GabaA
Neocortical Pyramidal Neuron: Deep	[-]	Input receptor
	Dap	GabaA
	Dam	GabaA
	Dad	GabaA
Ganglion Cells (Retina)	[-]	Input receptor
	Ded	GabaA
	Dem	GabaA
	Dep	GabaA
Olfactory Projection neuron	[-]	Input receptor
	Dem	GabaA
Neostriatal cholinergic interneuron	[-]	Input receptor
	Ded	GabaA

OntoQuery id:q1_3 Source Name:**Neuronal SO**

Specific neuron		
[-]	Neuronal structure	
medium spiny neuron: nucleus accumbens	[-]	Image
	dendritic spines	ACC1_2ma
Cerebellar Purkinje Cell	[-]	Image
	dendritic spines	pccor10_dc
Neostriatal Spiny Neuron	[-]	Image
	dendritic spines	oka4 osaka4 wt_g8T6

Figure 7. Federated ontological subqueries results. The information is transformed from the ontological database results using straight term mapping as well as mapping rules.

OntoQuery id:q1 Source Name:**Neuronal SO**

Neuron

[-]	Neuronal Compartment					Neuronal structure	
Neocortical Pyramidal Neuron: Superficial	[-]	Input receptor	Ionic conductance				
	Distal dendrite		[-]	peak conductance			
			gK2	0.1 mS/cm^2			
	Proximal dendrite		[-]	peak conductance			
			gK2	0.1 mS/cm^2			
	Dad	GabaA					
Olfactory Cortex Pyramidal Neuron	[-]	Input receptor	Ionic conductance				
	Dad	GabaA					
	Dap	GabaA					
Olfactory Bulb Mitral Cell	[-]	Input receptor	Ionic conductance				
	Dad	GabaA					
	Dam	GabaA					
	Dap	GabaA					
	Dbd	GabaA					
	Dbm	GabaA					
	Dbp	GabaA					
Cerebellar Purkinje Cell	[-]	Input receptor	Ionic conductance			[-]	Image
	Ded	GabaA				dendritic spines	pccor10_dc
	Dem	GabaA					
	Dep	GabaA					
CA1 Pyramidal Neuron	[-]	Input receptor	Ionic conductance				
	Dam	GabaA					
	Dap	GabaA					
	Dbd	GabaA					
	Dbm	GabaA					
	Dbp	GabaA					
Dentate granule cell	[-]	Input receptor	Ionic conductance				
	Ded	GabaA					
	Dem	GabaA					
	Dep	GabaA					
Thalamic Relay Neuron	[-]	Input receptor	Ionic conductance				
	Dem	GabaA					
	Dep	GabaA					
CA3 Pyramidal Neuron	[-]	Input receptor	Ionic conductance				
	Dam	GabaA					
Neocortical Pyramidal Neuron: Deep	[-]	Input receptor	Ionic conductance				
	Dap	GabaA					
	Dam	GabaA					
	Dad	GabaA					
Ganglion Cells (Retina)	[-]	Input receptor	Ionic conductance				
	Ded	GabaA					
	Dem	GabaA					
	Dep	GabaA					
Olfactory Projection neuron	[-]	Input receptor	Ionic conductance				
	Dem	GabaA					
Neostriatal cholinergic interneuron	[-]	Input receptor	Ionic conductance				
	Ded	GabaA					

Figure 8. The final federated query result. Data has been integrated from the three source databases, the terminology has been converted to the federated ontology and the constraints have been applied to remove unwanted information.

Performance is affected primarily by network/Internet latency—data are accessed at different geographic locations-and restricted system resources: each resource allocates limited CPU power to external clients to avoid overload. Therefore, the actual numbers may vary between runs.

Discussion

The contributions of this work are in demonstrating how:

- Ontologies can be used as the basis for configuring the query interface to a federated database so the choices of classes and properties, as well as the annotation of the output, can be dynamically configured to use the vocabulary preferred by a user.
- Mapping Rule Definitions and Rules can be used to specify equivalences (mappings) between objects in one database and objects in another.
- Such equivalences can be employed during federated database query.

Application of the Approach to Controlled Vocabularies: A SNOMED Example

A tabular representation of transformation-mapping rules, when used by curatorial-software front-ends to compositional vocabularies (or vocabulary compendia such as UMLS), can prevent inappropriate combinations of concepts during the postcoordination process. Specifically, if a given postcoordinated concept's structure is indicated as being based on a specific rule definition, the concept-composition user interface can dynamically limit selection/search to only the appropriate concepts for each placeholder in the definition. We discuss how well existing vocabularies meet the requirements of our model, or how they can be adapted to do so.

- *The need to categorize concepts*. The UMLS Semantic Net is a very simple first approximation to a class structure. In SNOMED, where concepts are not associated specifically with a semantic type, any suitable high-level/general concept may be designated as a class for the purpose of a rule definition, so that all of this concept's descendants in the concept hierarchy become class members.
- *The need to associate Properties with Concept Classes*: source vocabularies typically record property information as part of their data model-e.g., LOINC concepts have a property structure (analyte/system/recording method, etc)—but this is lost after translation to UMLS. There is no reason, however, why this information could not be preserved in future versions of UMLS as metadata in specially designated property tables that capture specific

aspects of the data models of individual source vocabularies.
- One may need to *add additional relationship types* to the vocabulary to specifically support rule representation, as described shortly.

The constraining rules for SNOMEDs for correct attribute usage are defined in prose form in Chapter 4 of the SNOMED User's Guide. However, *computable representations* of the prose rules do not exist in the SNOMED relationships table. The User's Guide defines *domain*—the hierarchy/family of concepts to which an attribute *applies*; *range*—the set of permissible values *allowed* for an attribute, and *defining attributes*—the permissible set of attributes that apply to a given hierarchy. For example, the *domain* of the attributes "Specimen Procedure" and "Specimen Substance" is the hierarchy "Specimen"; the *range* of "Specimen Procedure" is the hierarchy "Procedure" and the range of "Specimen substance" is the hierarchy "substance". The *defining attributes* for "Pharmaceutical/Biological Product" are "has-active-ingredient" and "has-dose-form". Note that the range may not be a single hierarchy, but a set of nonhierarchically related concepts. For example, the range of the attribute "laterality" is the set of concepts ["Side", "Right", "Left", "Right and left", "unilateral"].

To represent this computationally (Table 1), we would need to introduce the relationship types "aplies-to" (domain-of) and "is-limited-to" (range-of), which are not currently in SNOMED. Using these new relationship types, we represent the above prose rules computationally thus, using descriptive names instead of concept and relationship IDs:

Table 1 ■ Computable Representations of Prose Rules in SNOMED User's Guide

Concept1	Relationship	Concept2
Specimen-procedure	applies-to	specimen
Specimen-procedure	is-limited-to	procedure
Specimen-substance	applies-to	specimen
Specimen-substance	is-limited-to	substance
Laterality	is-limited-to	side
Laterality	is-limited-to	right
Laterality	is-limited-to	left
Laterality	is-limited-to	right and left
Laterality	is-limited-to	unilateral

In the table above, we have multiple rows for "Laterality", one for each permissible value, to preserve the SNOMED relational table structure. The set of permissible values for a given attribute is composed by a straightforward SQL statement, such as:

Select Concept2 from Relationships where Concept1="Laterality"

And Relationship = "is-limited-to"

A rule table such as the above can control a user interface for annotation of a new concept belonging to a specific class/hierarchy by (a) ensuring that only attributes belonging to the "defined attribute" set are presented to the user, by gathering all the "applies-to" attributes for that concept class; (b) when an attribute is chosen, constraining the selectable (or searchable) values of this attribute from the range of this attribute.

Implementation and Efficiency/Scalability Issues

We contrast our approach with OWL-based approaches toward ontology-based reasoning; OWL is a markup language with a fixed set of primitives, not a programming language arbitrarily extensible through a subroutine mechanism. Consequently, many types of rules (e.g., those involving mathematical expressions) cannot be expressed even in OWL-Full, the most expressive of the OWL variants. While OWL-Full allows the ontology designer to blur the distinction between classes and objects, at the cost of computational performance, such blurring is neither necessary nor desirable for the specific problem that OntoMediator addresses: RuleDefs must apply only to classes, and rules only to objects.

Our benchmark is dominated by network latency and the time required by individual CPUs to process SQL and XML: rule execution (i.e., their translation to SQL) is an insignificant component. This may not be true of different scenarios where the volume of data is much larger, but where the data are accessible on a single CPU—e.g., "vocabulary server" applications. We have reason to believe, however, that our approach will scale well.

- Our approach of translating/compiling rules into a stack-machine representation is analogous to the approach of the well-known compiler-generator utility *lex*,[43] which compiles regular expressions into a state-transition table: it allows significantly faster evaluation without having to reinterpret rule expressions each time a query is run.
- Relational database management systems (RDBMSs) are particularly suited to managing large vocabularies: the UMLS installation software even generates oracle scripts. RDBMS use allows the use of additional optimization mechanisms such as indexing of the individual columns used for internal rule storage, which would be useful if the rule set became large.

One should note that OWL itself relies on XML for its performance (or lack thereof), but XML itself was originally designed as data interchange mechanism rather than as a primary medium for manipulation of large data volumes or rule-bases. IBM has implemented an engine, the Integrated Ontology Toolkit,[44] using their RDBMS, DB2 (which includes an XML indexing engine) and Java. After import of OWL documents, the engine can achieve dramatically improved OWL inferencing performance compared to engines that operate on raw/textual XML.

Lessons Learned: Future Work

The implementation described is operational for SenseLab, a neuroscience site that receives about 900,000 hits per month. SenseLab's browsing-interface metaphor assumes minimal informatics sophistication on the user's part: the typical user is a neuroscientist, not a neuro-informatician. The ontology-mapping-rule infrastructure is invisible "plumbing" that is configured by an administrator-level person: the user accesses the resulting functionality through the equivalent of "canned queries" or stored procedures. The limitations that our users have conveyed to us mainly relate to domain-content coverage.

This does not imply that our model has no limitations, only that our user base appears far more concerned with accuracy and comprehensiveness of the information retrieved, and

has not been able to push the model to its limits. Our evaluation is therefore necessarily limited in scope, and describing our model in sufficient depth, with additional implementation details in the online appendix (available at www.jamia.org) to allow replication by others, will ultimately provide a better test of its robustness or lack thereof.

However, while it is always possible to achieve grain transformation in the database-federation scenario through custom, ad-hoc code, devising a transformation framework based on rules and generic code seemed to us to be more efficient to integrate data for our users. Such an approach improves long-term system maintainability, and allows the possibility of generalization to other related areas.

While we have shown that our model supports federated database queries, its applicability to controlled vocabularies has not been tested in the form of an actual implementation with a compositional vocabulary, and it is very likely that the model will need considerable augmentation before it can yield an operational system. Our planned future work will involve devise such an implementation for a neuroscience vocabulary to test and extend the ideas described here.

Availability of software: The existing Ontology Server code including the OntoMediator will be made freely available on request to the first author.

References ■

1. Sujansky W. Heterogeneous database integration in biomedicine. J Biomed Inform 2001;34(4):285–98.
2. Hass L, Schwarz P, Kodali P, et al. A system for integrated access to Life Science data sources. IBM Syst J 2001;40(2):489.
3. Josifovski V, Risch T. Query decomposition for a distributed object-oriented mediator system. Distributed Parallel Databases 2002;11(3):307–36.
4. Hartung J, Knapp G, Sinha B. Statistical Meta-analysis with applications, 2008.
5. McConnell S. Table-Driven methods. In: Code Complete. 2nd Edition. Redmond, WA: Microsoft Press; 2004.
6. White MD, Kolar LM, Steindel SJ. Evaluation of vocabularies for electronic laboratory reporting to public health agencies. J Am Med Inform Assoc 1999;6(3):185–94.
7. Regenstrief Institute. LOINC Home Page, 2008. Available at: http://loinc.org/. Accessed: Sept. 1, 2008.
8. College of American Pathologists. SNOMED clinical terms (SNOMED CT), 2008. Available at: http://www.snomed.org. Accessed: Sept. 2, 2008.
9. Nardi D, Brachman R. An introduction to description logics, 2003. Available at: http://www.inf.unibz.it/~franconi/dl/course/dlhb/dlhb-01.pdf. Accessed: Sept 1, 2008.
10. Paton NW, Stevens R, Baker PG, et al. Query processing in the TAMBIS bioinformatics source integration system. In: 11th International Conference on Scientific and Statistical Databases (SSDBM) Proc ed. 1999. IEEE Press: Los Alamitos: Cancer; 1999:138–47.
11. Stevens R, Baker P, Bechhofer S, et al. TAMBIS: Transparent access to multiple bioinformatics information sources. Bioinformatics 2000;16(2):184–6.
12. Mork P, Halevy A, Tarczy-Hornoch P. A model for data integration systems of biomedical data applied to online genetic databases. Proc AMIA Symp 2001:473–77.
13. Mork P, Shaker R, Halevy A, Tarczy-Hornoch P. PQL: A declarative query language over dynamic biological schemata. Proc AMIA Symp 2002:533–37.
14. Wheeler D, Barrett T, Benson D, et al. Database resources of the National Center for Biotechnology Information. Nucleic Acids Res 2007;35. (Database issue):D5–12.
15. OMIM. Online Mendelian Inheritance in Man. 2002. Available at: http://www.ncbi.nlm.nih.gov/omim/. Accessed: Nov 10, 2002.
16. Ashburner M, Ball CA, Blake JA, et al. Gene ontology: Tool for the unification of biology. The Gene Ontology Consortium. Nat Genet 2000;25(1):25–9.
17. Wang K, Tarczy-Hornoch P, Shaker R, et al. Data integration: beyond genomics to neuroscience data. AMIA Fall Syjmposium 2005; IEEE. Press, 2005.
18. Ludäscher B, Gupta A, Martone M, Model-Based. Information integration in a neuroscience mediator system. In: Proceedings of 26th International Conference on Very Large Databases. Cairo, Egypt, 2000; pp. 639–42.
19. Baru C, Gupta A, Ludäscher B, et al. XML-based information mediation with MIX. In: ACM Conference on Management of Data: A publication. 1999; Philadelphia, PA.
20. Oster S, Langella S, Hastings S, Ervin D, Madduri R, Phillips J, et al. caGrid 1.0: an enterprise Grid infrastructure for biomedical research. J Am Med Inform Assoc 2008;15(2):138–49.
21. Saltz J, Hastings S, Langella S, et al. A roadmap for CaGrid, an enterprise grid architecture for biomedical research. Stud Health Technol Inform 2008;138:224–37.
22. Kim W, Seo J. Classifying schematic and data heterogeneity in Multidatabase systems; IEEE. Computer 1991;24(12):12–8.
23. Cheung K, Nadkarni P, Miller P, Shin D. Automatic query mapping among genomic databases: A pilot exploration. Proc AMIA Symp 1998:942–6.
24. Bernstein P, Melnik S, Petropoulos M, Quix C. Industrial strength schema matching. SIGMOD 2007; Beijing, China.
25. Bernstein P, Melnik S, Churchill J. Incremental schema matching. VLDB 2006; Seoul, Korea.
26. Bernstein P, Melnik S. Model Management; 2:0: Manipulating Richer Mappings. In: SIGMOD 2007; Beijing, China.
27. Lam H, Marenco L, Clark T, et al. Integration of neurodegeneration data using RDF. BMC Bioinform 2007;8(Suppl 3):S4.
28. World Wide Web Consortium. OWL web ontology language: Overview, 2004. Available at: http://www.w3.org/TR/owl-features/. Accessed: Sept 1, 2005.
29. Kohler J, Philippi S, Lange M. SEMEDA: Ontology based semantic integration of biological databases. Bioinformatics 2003;19(18):2420–7.
30. Wikipedia. Ontology (definition in computer science). Available at: http://wikipedia.org/wiki/Ontology_(computer_science). Accessed: Mar 23, 2008.
31. Marenco L, Wang T, Shepherd G, et al. A Framework for Biomedical Database Federation. J Am Med Inform Assoc 2004;11(6):523–34.
32. Bowden D, Martin RF. NeuroNames brain hierarchy. Neuroimage 1995;2(1):63–83.
33. Lindberg DAB, Humphreys BL, McCray AT. The unified medical language system. Methods Inform Med 1993;32:281–91.
34. Masys D. An evaluation of the source selection elements of the prototype UMLS information sources Map. Proc Annu Symp Comput Appl Med Care;1992:1992,295–8.
35. Shepherd G, Mirsky JS, Healy MD, et al. The human brain project: Neuroinformatics tools for integrating, searching and modeling multidisciplinary neuroscience data. Trends Neurosci 1998;21(11):460–8.
36. Miller P, Nadkarni P, Singer M, et al. Integration of multidisciplinary sensory data: A pilot model of the human brain project approach. J Am Med Inform Assoc 2001;8(1):34–48.
37. Martone ME, Zhang S, Gupta A, et al. The cell-centered database: A database for multiscale structural and protein localiz

ation data from light and electron microscopy. Neuroinformatics 2003;1(4):379–95.
38. Martone ME, Tran J, Wong WW, et al. The cell centered database project: An update on building community resources for managing and sharing 3D imaging data. J Struct Biol 2008;161(3):220–31.
39. Dyhrfjeld-Johnsen J, Maier J, Schubert D, et al. CoCoDat: A database system for organizing and selecting quantitative data on single neurons and neuronal microcircuitry. J Neurosci Methods 2005;141(2):291–308.
40. Aho AV, Sethi R, Ullman JD (Compilers). Principles, Techniques, Tools. Reading, MA: Addison-Wesley, 2003.
41. Marenco L, Tosches N, Crasto C, et al. Achieving evolvable web-database bioscience applications using the EAV/CR framework: Recent advances. J Am Med Inform Assoc 2003; 10(5):444–53.
42. Nadkarni PM, Marenco L, Chen R, et al. Organization of Heterogeneous Scientific Data Using the EAV/CR representation. J Am Med Inform Assoc 1999;6(6):478–93.
43. Levine JR, Mason T, Brown D. lex. & yacc. Sebastopol, CA: O'Reilly & Associates, 1992.
44. IBM Corporation. IBM Integrated Ontology Development Toolkit, 2004. Available at: http://www.alphaworks.ibm.com/tech/semanticstk/faq. Accessed: Mar 2, 2009.