Ontology Database: A New Method for Semantic Modeling and an Application to Brainwave Data

Paea LePendu[1], Dejing Dou[1], Gwen A. Frishkoff[2], and Jiawei Rong[1]

[1] Computer and Information Science
University of Oregon, USA
{paea,dou,jrong}@cs.uoregon.edu
[2] Learning Research and Development Center
University of Pittsburgh, USA
gwenf@pitt.edu

Abstract. We propose an automatic method for modeling a relational database that uses SQL triggers and foreign-keys to efficiently answer positive semantic queries about ground instances for a Semantic Web ontology. In contrast with existing knowledge-based approaches, we expend additional space in the database to reduce reasoning at query time. This implementation significantly improves query response time by allowing the system to disregard integrity constraints and other kinds of inferences at run-time. The surprising result of our approach is that load-time appears unaffected, even for medium-sized ontologies. We applied our methodology to the study of brain electroencephalographic (EEG and ERP) data. This case study demonstrates how our methodology can be used to proactively drive the design, storage and exchange of knowledge based on EEG/ERP ontologies.

1 Introduction

With recent advances in data modeling and increased use of the Semantic Web, scientific communities are increasingly looking to ontologies to support web-based management and exchange of scientific data. Ontologies can be used to formally specify concepts and relationships between concepts within a domain. The resulting logic-based representations form a conceptual model that can help with storage, management and sharing of data among different research groups.

In addition to the representation of classes and properties, ontologies can store intensional knowledge in the form of general facts, often called rules, axioms or formulae, such as, "All Sisters are Siblings." Extensional data include specific facts, or ground terms, such as, "Mary and Jane are Sisters." Relational databases can effectively store and retrieve extensional data, but they lack obvious mechanisms to perform the inferences necessary to answer extensional queries over intensional data, as in, "Which individuals are Siblings?" Unlike a typical relational database, a knowledge base can support the deduction that Mary and Jane are siblings by using an inference engine.

B. Ludäscher and Nikos Mamoulis (Eds.): SSDBM 2008, LNCS 5069, pp. 313–330, 2008.

Intensional knowledge reduces the need to store large amounts of extensional data. For example, we do not need to store the fact, "Mary and Jane are Siblings," to know that it is true. The trade-off, however, is that inferences are required at run-time to generate this fact. What we have, therefore, is an example of the classical trade-off between time and space: the more extensional data we store, the less time it will take to answer queries about them. In this paper, we challenge traditional approaches for modeling knowledge-based or deductive database systems of this sort, which typically aim to find a balance between space and time requirements. Instead we propose that space is expendable and a great deal of inference (time) can be saved through the use of triggers and foreign-keys to forward-propagate inferences at load-time. Interestingly, when we compared our methods against existing benchmarks, we found we significantly improved query performance as expected, but load-time was remarkably unaffected.

In addition to these performance gains, we demonstrate that semantics can play an essential role in data management and query answering. In fact, both ontologies and database systems are important, leading us to propose a new methodology for database design, which we will call *ontology databases*.

To illustrate this idea, we describe the application of our methodology to brain electroencephalographic (EEG and ERP) data. In this application, we describe a database design that is ontology-driven. Moreover, we demonstrate how queries can be posed by domain experts at the ontology-level rather than using SQL directly. Database projects like ZFIN [8] and MGI [1], housing large central repositories for zebrafish and mouse genetic data, respectively, were later reinforced by the Gene Ontology [25] to help normalize knowledge across these kinds of repositories. By contrast, our Neural ElectroMagnetic Ontology (NEMO) project uses expert knowledge in the form of EEG/ERP ontologies to drive the data modeling and information storage and retrieval process.

The paper is organized as follows. We begin with related work (Section 2), followed by a description of our ontology-based modeling methodology and a performance analysis (Section 3). We then present a case study in which we applied our methodology to develop ontology databases for EEG/ERP query answering (Section 4). We conclude with a discussion and an outline of future work in Section 5.

2 Related Work

Ontologies can be regarded as a conceptual or semantic model for database design. Hull and King [19] provide a nice summary of semantic models of all kinds: Entity-Relational, Object-Oriented, Ontological and so on. While the notions in their survey make clear that there are firm connections between models, database implementations, and logics, we have been interested in exploring the question, "What is a semantic data model?" In particular, we wish to explore it from an ontology-based perspective that addresses practical issues in collaborative scientific research, especially, biomedical research. Increasingly, biomedical researchers are looking to develop ontologies to support cross-laboratory data

sharing and integration. These ontologies can be found at ontology repositories around the world [34]. For example, more than 62 biomedical ontologies can be found at the National Center for Biomedical Ontology (NCBO) [6].

Pan and Heflin proposed a similar approach, which they call description logic databases (DLDB) [26]. DLDB is a storage and reasoning support mechanism for knowledge base facts (RDF triples), which has been compared to well-known systems such as Sesame [10]. Although we structure the database relations in a way that is similar to DLDB (i.e., unary and binary predicates become unary or binary relations), our implementation using triggers and foreign keys to support reasoning, as opposed to SQL views, allows for a significant performance gain by trading space for time by eagerly forward-propagating data at load-time. In this context, it is informative to consider the recent work by Paton and Díaz [27], which examines rules and triggers in active database systems.

Recent research on bridging the gap between OWL and relational databases by Motik, Horrocks and Sattler [24] provides unique insight into the expressiveness of description logics versus relational databases. The integrity constraints in databases can be described with extended OWL statements (axioms). An important contribution of this research is to show that the constraints can be disregarded while answering positive queries, if the constraints are satisfied by the database.

The idea of balancing space and time when we couple databases and reasoning mechanisms comes from seminal works by Reiter [28,30]. Reiter proposed a system that uses conventional databases for handling ground instances, and a deductive counterpart for general formulae. Since no reasoning is performed on ground terms, Reiter argues convincingly that in such a system queries can be answered efficiently while retaining correctness. OntoGrate [13] is precisely such a system for semantic query translation using ontologies. The key question that motivated our trigger-based approach was, "Since disk-space is rarely an issue these days, what would happen if we use even more space?"

The neuroscience community is a recognized leader in the development of biomedical ontologies. For example, the Human Brain Project has supported the development of a common data model and meta-description language [17] for neuroscience data exchange and interoperability. BrainMap [22] has designed a Talaraich coordinate-based archive for sharing and meta-analysis of brain mapping studies and literature, as well as a sharable schema for expression of cognitive-behavioral and experiment concepts. The fBIRN project [20] has pioneered several areas for neuroscience data sharing, including distributed storage resources and taxonomies of neuroscience terms (called BIRNlex). Our project will build on this prior work and extend it to incorporate ontology-based methods for reasoning. In addition to incorporating cognitive-behavioral and anatomy concepts represented in BrainMap and in fBIRN, NEMO will develop ontologies for temporal, spatial, and spectral concepts that are used to describe EEG and ERP patterns. In line with OBO "best practices," we will reuse ontology concepts from relevant domains. In fact, we are collaborating directly with ontology engineers and domain experts in the fMRI, as well as the EEG and ERP, communities.

The NEMO project brings some distinctive methods to bare on the problem of data sharing. Whereas most prior work on data sharing in the neurosciences has focused on the development of simple taxonomies or relational databases, NEMO uses ontologies to design databases that can support semantically based queries. What this means is that NEMO databases can be used to answer more complex queries, which cannot be handled by traditional (purely syntactic) database structures. For example, the popular Gene Ontology (GO) [25] provides a standard vocabulary and concept model for molecular functions, biological processes and cellular components in genetic research. The OWL [7] specification of GO is over 40 Megabytes in size [25] and terabytes of research data stored in model organism databases around the world such as ZFIN [8] and MGI [1] are all being marked-up according to the GO ontology. The NEMO working group is borrowing from this idea and taking it a step further [12,15]. More than a standard vocabulary of terms, the ontologies NEMO is developing will capture knowledge ranging from the experimental methods used to gather ERP data down to instrument calibration settings so that results can be shared and interpreted semantically during large-scale meta-analysis across laboratories.

3 Ontology-Based Data Modeling

We first present a new and general methodology, which takes a Semantic Web ontology as input and outputs a relational database schema. We call such a database an "ontology database," which is an ontology-based, semantic database model. As we will show in Section 4, after we load ERP data into the NEMO ontology database, we can answer queries based on the ontology while automatically accounting for subsumption hierarchies and other logical structures within each set of data. In other words, the database system is ontology-driven, completely hiding underlying data storage and retrieval details from domain experts, whose only interaction-interface happens at the ontology (conceptual) level.

3.1 The Procedural Extension

Although Description Logics (DL) [9] provide the formal logical foundation for OWL and Semantic Web ontologies, we do not require the full expressiveness of this logic for data modeling purposes in most scenarios we have encountered. It suffices to use rules of the form (reads "if C then D"):

$$C \Rightarrow D,$$

which exclude the analysis-by-cases and contrapositive reasoning provided by full DL inclusion axioms of the form (reads "C is subsumed by D"):

$$C \sqsubseteq D.$$

What this means is that we are drawing a line between databases and knowledge bases. For example, while it may be taken for granted in a knowledge-based

system that, "X is either a Rock or it is not a Rock, no matter what X is," a database has no such reasoning capability. It can only say which is actually the case. As such, we technically only allow epistemic inclusion axioms with the **K** operator [9] which stands for "know" in the following rule (reads "Only when we *know* that C is true can we conclude D"):

$$\mathbf{K}C \sqsubseteq D.$$

The difference is evidenced by the fact that we can immediately conclude D (without any positive or negative witnesses of C) in:

$$(C \sqcup \neg C) \sqsubseteq D,$$

but not necessarily in:

$$(\mathbf{K}C \sqcup \mathbf{K}\neg C) \sqsubseteq D.$$

This restriction makes knowledge maintenance (reasoning) much easier: all we need to calculate is the *procedural extension* of a given set of facts and rules [9]. This can easily be done using database triggers and foreign keys with cascading deletes, the basic idea of which we outline below.

3.2 Triggers

Triggers are used for each rule to propagate data in a forward-chaining manner as facts are loaded into the ontology database. For example, suppose we have the following first-order rule (reads "all Sisters are Siblings"):

$$\forall x, y: \ Sisters(x, y) \rightarrow Siblings(x, y).$$

Whenever a new pair of sisters is inserted into the ontology database, such as $Sisters(Mary, Jane)$, a trigger fires, eagerly inserting $Siblings(Mary, Jane)$ as well. This process is depicted in Figure 1.

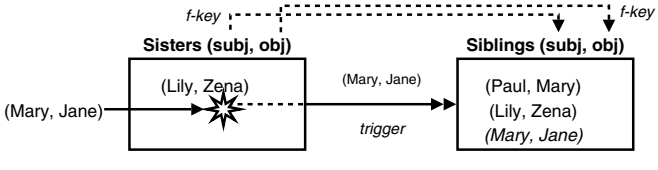

Fig. 1. This figure shows that upon asserting $Sisters(Mary, Jane)$ which means inserting $(Mary, Jane)$ into the Sisters-property table, the trigger causes $(Mary, Jane)$ to first be inserted into the Siblings-property table. Triggers generate knowledge in a forward-chaining manner for the Sisters-Siblings rule, $\forall x, y : \ Sisters(x, y) \rightarrow Siblings(x, y)$. Implicitly understood in this sub-property rule is also the contrapositive, $\forall x, y: \ \neg Siblings(x, y) \rightarrow \neg Sisters(x, y)$, an integrity check that foreign-keys can enforce, shown here as the dotted line.

Although the above is an example of a sub-property (Sisters is a sub-property of Siblings), triggers can be used for both sub-class and sub-property hierarchies. Each trigger is a straightforward encoding of the epistemic rule, in SQL:

```
CREATE TRIGGER subPropertyOf-Sisters-Siblings SUCH THAT
UPON DETECTING EVENT INSERT (x,y) INTO Sisters(subject,object)
FIRST EXECUTE INSERT (x,y) INTO Siblings(subject,object)
```

3.3 Foreign Keys with Cascading Delete

Foreign keys are used to check integrity constraints as usual, but by using the "on delete cascade" option, they also propagate deletions whenever facts are negated (which is not uncommon in scientific domains). For example, in the Sisters-Siblings sub-property rule of Figure 1 it is understood implicitly that if two people are *not* Siblings, then they cannot be Sisters either:

$$\forall x, y : \ \neg Siblings(x, y) \rightarrow \neg Sisters(x, y).$$

Semantically, we interpret the contrapositive to mean two things. First of all, it is an integrity constraint: if $Siblings(Mary, Jane)$ is not true, then it cannot be the case that $Sisters(Mary, Jane)$ is true, so an integrity check is performed to validate that $Siblings(Mary, Jane)$ is true before inserting $Sisters(Mary, Jane)$. Of course, care must be taken to ensure triggers and integrity checks happen in the correct order (note the "FIRST" keyword in the SQL trigger). Secondly, if deletions (negations) are performed, they must be propagated to ensure consistency is maintained, thus explaining the "on delete cascade" option. Indeed, this is the pattern for all sub-class and sub-property rules: they are both triggers (knowledge generating) and integrity constraints (knowledge checking), consistent with the semantics of inclusion axioms.

Integrity constraints also occur in domain and range restrictions on properties. In this case, we have foreign keys but no triggers. For example, when we assert $Sisters(x, y)$ we generally presume that x and y are People. That is, we mean:

$$\forall x, y : \ [\neg Person(x) \cup \neg Person(y)] \rightarrow \neg Sisters(x, y),$$

but not necessarily:

$$\forall x, y : \ Sisters(x, y) \rightarrow [Person(x) \cap Person(y)].$$

In other words, given the statement $Sisters(Mary, buddyTheFrog)$, we do not intend to automatically conclude that $buddyTheFrog$ is a Person but rather hope the assertion is rejected unless we know for sure that $buddyTheFrog$ is a Person (and not a Frog). This kind of reasoning is due in large part to the notion common in database systems that any fact not known to be true is presumed false, known as the *closed world assumption* [29].

Table 1. The *ontology database* methodology is summarized in this table. Here, respectively, *subj* and *obj* refer to the subject and object of a property, *MinCard* and *MaxCard* refer to cardinality, and *f-key* and *p-key* stand for foreign key (with an "on delete cascade" option) and primary key.

Logical Feature	FOL Formalism	Ontology DB Implementation
Structure		
$Class(A), Class(B)$	$A(x), B(y)$	relation: $A(id), B(id)$
$Property(P)$	$P(x, y)$	relation: $P(subj, obj)$
Restrictions		
$Domain(P, A)$	$\forall x, y : P(x, y) \rightarrow A(x)$	f-key: $P(subj)$ ref $A(id)$
$Range(P, B)$	$\forall x, y : P(x, y) \rightarrow B(y)$	f-key: $P(obj)$ ref $B(id)$
$MaxCard(P, 1)$	$\forall x, y, z : P(x, y) \wedge P(x, z) \rightarrow y = z$	p-key: $P(subj)$
$MinCard(P, A, 1)$	$Domain(P, A) \rightarrow (\forall x : A(x) \rightarrow \exists y : P(x, y))$	f-key $P(subj)$ ref $A(id)$; trigger: on insert on A(id) insert ignore $P(id, null)$
Subsumption		
$subClassOf(B, A)$	$\forall x : B(x) \rightarrow A(x)$	trigger: **before** insert on B(id) insert ignore $A(id)$; f-key: $B(id)$ ref $A(id)$;
$subPropertyOf(Q, P)$	$\forall x, y : Q(x, y) \rightarrow P(x, y)$	trigger: **before** insert on Q(subj,obj) insert ignore $P(subj, obj)$; f-key: $Q(subj, obj)$ ref $P(subj, obj)$;
	Horn Rules & GMP	
	$\forall x_1, x_2 \ldots x_m :$ $P_1(x_1, x_2) \wedge \ldots \wedge P_n(x_{m-1}, x_m) \rightarrow Q(x_i, x_j)$ $(1 \leq i, h \leq m, 1 \leq j, h \leq m)$	$\forall k \in [1..n]$ trigger(rule_premise-k): on insert on $P_k(x_{h-1}, x_h)$ update [rule-premise-table with P_k] trigger(rule_activate): on update on [rule-premise-table] if [all premises satisfied] then insert ignore $Q(x_i, x_j)$ $(1 \leq i, h \leq m, 1 \leq j, h \leq m)$

3.4 Modeling Summary

Table 1 summarizes the main logical features we implement in the ontology database methodology. These features can be categorized according to structures, restrictions and subsumptions which come from OWL, RDF [3] and general first-order logic. The database relational structure we have chosen (unary and binary predicates become unary and binary relations) is almost identical to the hybrid approach of DLDB [26], which combines approaches from prior works to effectively store RDF triples.

3.5 Logical Justification

Our ontologies are generally restricted to Horn Normal Form (HNF) [32], which is a disjunction with only one positive literal as in:

$$\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \vee q.$$

These formulae can be written as implications without disjunctions on the right-hand side, like Datalog [33] rules, which we call implicative normal form (INF):

$$p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q.$$

Generalized Modus Ponens (GMP) [32] is an inference rule based on the well-known modus ponens rule:

$$\frac{p'_1 \wedge p'_2 \wedge \ldots \wedge p'_n \quad p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q}{SUBST(\theta, q)}\ GMP$$

GMP allows us to unify several antecedents simultaneously to prove a conclusion. It is well-known that GMP is sound and complete for knowledge bases in HNF (and therefore INF) [32]. A trigger is essentially a forward-chaining implementation of GMP, recursively calling other triggers as necessary. Because all definitions are acyclic, the procedure is guaranteed to terminate. Foreign-keys and null-valued triggers together provide the machinery for solemnization under existential constraints (such as, "All Employees have an SSN." [31]). According to this method, an ontology database therefore produces and maintains the procedural extension, guaranteeing that the database is a Herbrand Model for the given set of facts (see [32] for details on the Herbrand universe, interpretation and model).

3.6 General Performance Analysis

We tested our methodology using the Lehigh University Benchmark (LUBM) [18] ontology[1], and compared the load-time (see Figure 2) and query-answering (see Figure 3) performance against DLDB [26], an ontology data storage model not unlike our own.

The LUBM features an ontology for the university domain (e.g., faculty, courses, departments, etc.) together with a data generation tool for creating OWL datasets of arbitrary size and a set of queries for evaluating performance. The most significant difference between DLDB and our ontology database (OntoDB) is that DLDB uses SQL views instead of triggers to propagate subsumptions. In other words, our approach is like an *eager* evaluation strategy for subsumption inferences whereas DLDB is *lazy*. Because we propagate knowledge as data is loaded so as to increase query performance, we expected to incur a load-time hit. To our surprise, the load-time was largely unaffected even though query-time benefitted significantly. Our only explanation of this phenomenon is that the underlying database file system is optimized to perform several insertions (caused by triggers) in relatively constant-time – which might eventually be affected as the depth of the subsumption hierarchy grows. Naturally, our approach uses more disk-space (roughly 3-times the space), a trade-off we knew we had to make (space versus time has to give) [30]. Again, our results are summarized in Figures 2 and 3.

[1] All experiments were performed on an unremarkable personal laptop computer with a 1.8Ghz Centrino processor and 1Gb of RAM running MySQL 5.0 as the RDBMS.

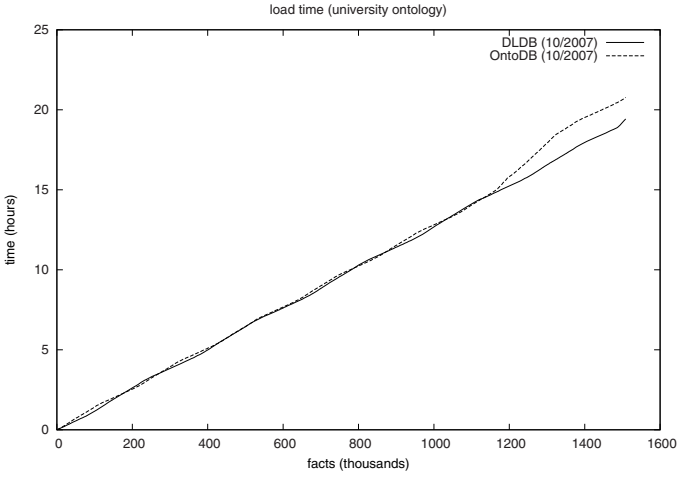

Fig. 2. The load-time results for the Lehigh University benchmark ontology data show that the load-time of our ontology database approach (OntoDB) is comparable to that of DLDB. The blips at around 1.2M and 1.4M are probably due to disk resizing or other background effects.

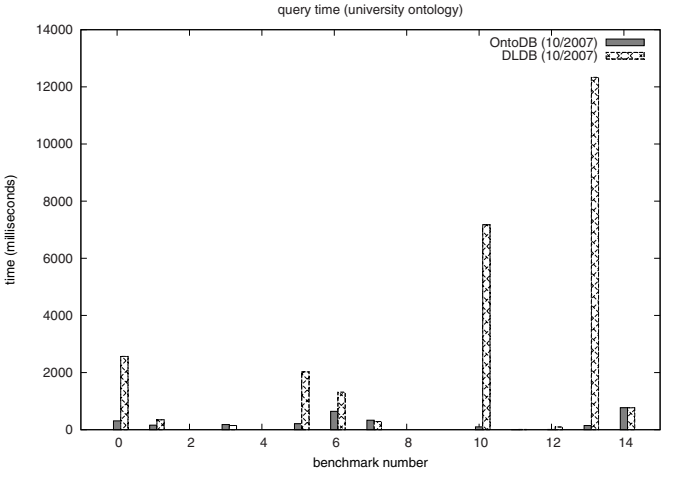

Fig. 3. The query-answering time results for the Lehigh University benchmark ontology data show that the query-time of our ontology database approach is often significantly faster than DLDB. Charted here are the running-times for 10 of the 14 benchmark queries published by Lehigh University. Queries 2, 4, 8, and 9 are not shown here due to scale, having extremely long running-times.

4 Case Study: Application of Ontology Databases to Brainwave Data

4.1 EEG and ERP Data Sharing in Clinical and Cognitive Neuroscience

The problem of data sharing in brain electromagnetic research, like that in other scientific fields, is challenged by data scale, multivariate parameterizations, and dimensionality. Neuroinformatics must address these challenges with robust data management and integration techniques. To this end, neuroinformatics researchers have developed a number of database and XML-based methods [21], providing effective solutions for annotation and storage of complex, large-scale datasets. Going beyond syntax and structure, the hard problems that remain are closely linked to the neuroscience community's requirements for rich semantic representation and integration of patterns across disparate experiment and laboratory procedures and paradigms.

The development of ontologies may be central to addressing these problems. Indeed, adoption of ontologies has already enabled major scientific progress in biomedical research [6,20,23,25] and is a rapidly growing area in bioinformatics and neuroinformatics research. The present work aims to extend and combine Semantic Web and database modeling technologies to address issues in ERP data representation and semantic query answering. The project is called "Neural ElectroMagnetic Ontology" (NEMO). Eventually, we hope that our ontology-based framework will support large-scale semantic data sharing, give rise to meta-analysis, and lead to major advances in brain functional mapping using ERP and related methods.

Electroencephalographic (EEG) data consist of changes in neuroelectrical activity measured over time (on a millisecond timescale), across two or more locations, using noninvasive sensors ("electrodes") that are placed on the scalp surface. A standard technique for analysis of EEG data involves averaging across segments of data ("trials"), time-locking to stimulus "events," to create *event-related brain potentials (ERPs)*. The resulting measures are characterized by a sequence of positive and negative deflections across time, at each sensor. For example, to examine brain activity related to language processing, the EEG may be recorded during presentation of words versus non-words, using 128 or more sensors (Figure 4). Averaging across trials within a given stimulus category accentuates brain activity that is related to processing the specific type of stimulus. In principle, activity that is not event-related will tend towards zero as the number of averaged trials increases. In this way, ERPs provide increased *signal-to-noise (SNR)*, and thus increased sensitivity to functional (e.g., task) manipulations.

The resulting datasets comprise rich sets of spatial, temporal, and functional (task-related) measurements. This case study describes the ontology that has been developed by domain experts and refined by data mining techniques to capture this knowledge. Furthermore, we demonstrate how the ontology database methodology can be used to automatically implement an effective storage and

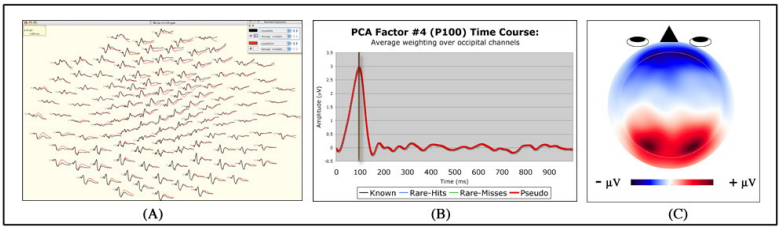

Fig. 4. (A) 128-channel EEG waveplot; positive voltage plotted up; responses to words versus non-words. (B) Time course of P100 factor for same dataset, extracted using Principal Components Analysis. (C) Topography of P100 factor (negative on top and positive at bottom). See [15] for details.

retrieval mechanism for ERP data that preserves the meaning and interpretation prescribed by domain experts.

4.2 ERP Ontology Development

In previous work, an ERP ontology for a limited domain (word recognition) was designed collaboratively with domain experts, using data collected in a series of visual word recognition experiments (see [12,16] for details). To support the development of an initial ERP ontology, based on automated data analysis and labeling, we applied data decomposition methods to help separate signal (brain activity) from noise (noncerebral artifacts) and to disentangle overlapping patterns [16]. More specifically, temporal Principal Components Analysis (PCA) was applied to ERP data consisting of 128 electrodes, 275 timepoints (sampling rate, 250Hz), 34 human subjects, and 4 experimental conditions (see [11] for details on PCA methods).

For each PCA factor, we extracted summary metrics representing spatial, temporal and functional dimensions of the ERP patterns of interest. Thus, the data represent the individual PCA factors, weighted across individual subjects and experiment conditions. These data were post-processed by ERP domain experts and represented as points in a 25 dimensional attribute space. In previous work, we characterized eight types of robust patterns, P100, N100, N2, N3, MFN, P1r/P2, N4 and P300 [16]. Rules for each pattern were based on results from prior literature. For example, the P100 rule was operationalized as follows:

$$\begin{aligned}\forall x, i, j: \; & [\, PCA\_Factor(x) \wedge (80 < i) \wedge ti\_max(x, i) \wedge (i < 150) \wedge \\ & factorEvent(x, STIMON) \wedge factorModality(x, VISUAL) \wedge \\ & in\_mean\_roi(x, j) \wedge (0 < j) \wedge roi(P100v, OCC)\,] \\ & \quad \rightarrow occursIn(P100v, x)\end{aligned}$$

where "ti_max" is the peak latency, "in_mean_roi" is the mean amplitude over a given region-of-interest (ROI), and ROI for "P100v" is specified as "occipital." In addition to "top-down" (expert-defined) pattern rules, we performed "bottom-up" (data mining) analysis using clustering-based classification to

discover class and property hierarchies and association rule mining to find axioms as a way to complement and refine the concepts and rules articulated by domain experts [12,16]. Evaluation was performed against a "Gold Standard" labeled dataset described in [16].

Our initial ERP ontology consists of classes, class taxonomy, properties and their relationships. The ontology consists of roughly 29 classes, 40 properties, 27 sub-class relationships, and 3 super-properties. We show a partial view of the ERP ontology in Figure 5. We would like to stress that this ontology has undergone significant changes since the time of this writing. The latest version of the NEMO ERP ontology is available online at `http://aimlab.cs.uoregon.edu/NEMO/NEMO_ERP.owl`[2] and will soon be available on NCBO. Figure 5 shows

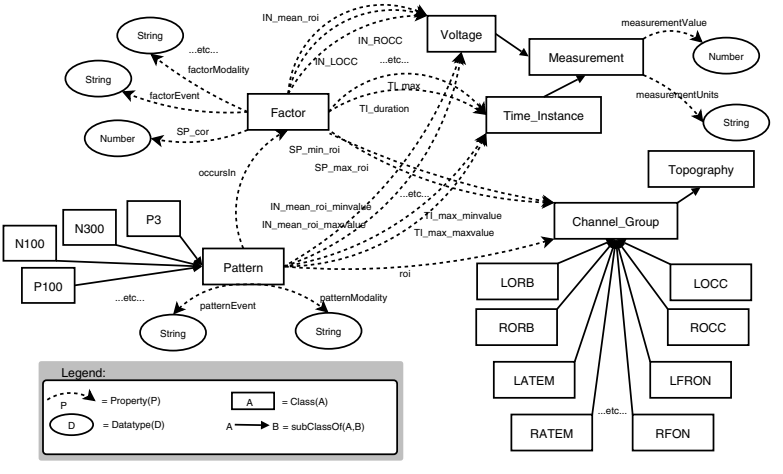

Fig. 5. A partial view of the ERP ontology

five basic classes, i.e., factor, pattern, channel group, topography and measurement. Factor objects have temporal, spatial and functional attributes (part of which are listed in the graph, such as factorEvent, SP_cor and factorModality) which are represented as properties of the factor class in the ERP ontology. TI-max and IN-mean(ROI) are properties of factor which relates the measurements (e.g., time-instance and voltage) which have both unit and value properties. The pattern class has 8 sub-classes (P100, N100, etc.) which correspond to the 8 ERP patterns defined by domain experts [16,12]. The properties of the pattern class are those used in expert rules or rules discovered by data mining. The expert rules are represented as Horn rules whose body are conjunctions of predicates. The relationship between factor and pattern can be modeled using the "occursIn" property. Each pattern has a region of interest, which is a channel group belonging to the topography class. Each area on the scalp can be divided into a left and right part. For instance, left occipital (LOCC) and right occipital

[2] Human readable OWLDoc: `http://aimlab.cs.uoregon.edu/NEMO/OWLdoc_ERP`

(ROCC) are sub-classes of channel group and the combination of them is called occipital (OCC) (not shown). The mean intensity (measured in microvolts) for each region of interest is calculated based on this relationship.

While the graph representation helps convey the general idea, we use a formal, first-order ontology language to represent the ontology internally. This internal language is (and has been) easily translated to and from standard ontology languages such as OWL [7] or OBO [2] for terminological knowledge and SWRL [5] for general Horn rules. We plan to contribute our ERP ontology to the National Center for Biomedical Ontology [6].

4.3 ERP Data Modeling Results

We applied our modeling methodology to the ERP ontology depicted in Figure 5 to investigate several properties: correctness, space, load-time, and query answering speed.

We worked with a visual word study data set in which there were 34 different human subjects, 25 different dimensions in the attribute space and a vector of 1152 different component factors after PCA decomposition. In essence, we were working with a relatively small matrix of data that was approximately 1152 rows by 30 columns in size.

For every class in the ERP ontology, we define a unary relation and for every property a binary relation. For every logical rule in the ontology specification, we generated the corresponding foreign keys, triggers, and primary keys in the database. Finally, for every data instance, we generated a unique internal object identifier. Altogether, the data essentially consists of 100,425 individual facts.

It took approximately 14 seconds to generate the database schema based on the ontology and load it into the MySQL RDBMS. It took 1.3 hours to load all of the individual facts. The entire ERP ontology database occupies roughly 10 MB of disk space, and contains over 145,000 facts (including new ones after all triggers). There are 29 tables for class concepts and 40 tables for properties. The class hierarchy has a depth of at most 5. The top-class in the hierarchy has 23,093 instances whereas the average-sized class has 1,152 instances. The ontology database generated 27 different triggers and 95 foreign-key constraints to maintain the procedural extension.

Figure 6 shows a visual representation of the entity-relation (ER) diagram for the ERP ontology database. Although too large and complex to show every detail in this paper, the diagram gives a rough idea of how many concepts (boxes) and dependencies (lines) are managed by the database (triggers are not shown).

As for query processing, we tested four different queries that exhibited the various properties of interest for our implementation: subsumption, data size (amount of data, joins, etc.), aggregate computations, and ease of formulation based on ontology concepts. A summary of the queries and properties they are meant to explore is shown in Table 2. Although we hoped to find at least some interesting and significant variations in query speed or formulation difficulty (according to domain experts), this was not the case. Each query proved extremely straightforward to formulate in SQL, and execution time was statistically unmea-

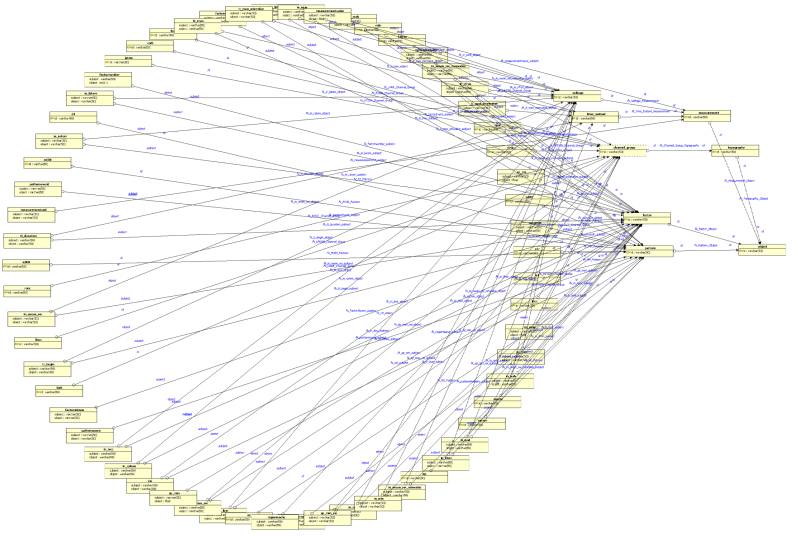

Fig. 6. Although too difficult to read in printed form, this visualization of the ER Diagram for the ERP ontology database gives a general sense of number and complexity of the concepts (boxes) and foreign-key relationships (lines). Central concepts such as "pattern," "factor," and "channel group" are toward the right-side of the image – they are the most densely connected nodes.

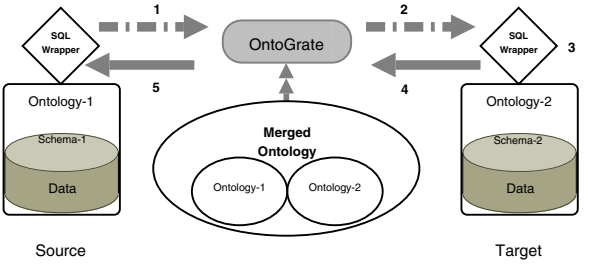

Fig. 7. A data integration scenario in which the user (1) issues a query using the semantics of the source ontology which (2) gets translated into the semantics of the target ontology using the inference engine in OntoGrate which is then (3) issued as a database query using a SQL syntax wrapper from which (4) target data is returned and finally (5) translated back into the source semantics for the user to interpret.

surable (somewhere between 0-40 ms) on our equipment. All answers returned by the database were 100% complete and sound (perfect recall and precision) as compared to answers expected by our domain experts. The answers and execution times for each query are also shown in Table 2. We would like to note that, although not the focus of this paper, the ontology database approach we describe adds the unique advantage that queries can be posed at the ontology-level

Table 2. This table lists the queries and answers verified by experts. Each query is meant to test various properties of interest.

Query / Answer	Property of Interest

(1) *Show the region of interest for all ERP patterns that occur between 0 and 300ms.*
 	
`Pattern ROI max_value min_value`
`==========`
`N100 LOCC 229 151`
`N100 ROCC 229 151`
`N2 LPTEM 300 230`
`N2 LOCC 300 230`
`P100 ROCC 150 60`
`P100 LOCC 150 60`
 	
`[Fetch MetaData: 0/ms] [Fetch Data: 10/ms]`
`[Execution: 0/ms]`
	subsumption, data size

(2) *Which PCA factor do P100 patterns most often appear in?*
 	
`Pattern occurances Factor_Number`
`==========`
`P100 133 4`
 	
`[Fetch MetaData: 0/ms] [Fetch Data: 0/ms]`
`[Execution: 20/ms]`
	subsumption, aggregation, data size

(3) *What is the range of intensity mean for the region of interest for N100 patterns?*
 	
`Pattern in_mean_roi_min in_mean_roi_max`
`==========`
`N100 -infinity -0.4`
 	
`[Fetch MetaData: 0/ms] [Fetch Data: 0/ms]`
`[Execution: 10/ms]`
	ease of formulation

(4) *Show the patterns whose region of interest is left occipital and occurs between 220 and 300ms.*
 	
`Pattern ROI max_value min_value`
`==========`
`N2 LOCC 300 230`
 	
`[Fetch MetaData: 0/ms] [Fetch Data: 0/ms]`
`[Execution: 10/ms]`
	subsumption, aggregation

by domain experts using languages such as SPARQL [4] or OWL-QL [14] and automatically translated to SQL using wrappers (see Figure 7).

5 Discussion and Future Work

In this paper, we have outlined a new framework for designing and implementing ontology databases. We have further presented a case study in which we applied

our method to ERP data. This ontology-driven data modeling approach appears promising, working well for: (1) scientific application scenarios requiring rich semantics, and (2) "query-mostly" scenarios common to such domains in which large sizes of data must be queried and analyzed significantly more often than data are loaded. We have also argued that the ontology database methodology using triggers and foreign-keys is logically justified: that it correctly generates and maintains a logical model for a given ontology and set of data.

In terms of scalability, the LUBM is fairly complex but medium in size. GO, on the other hand, has over 36,000 different concepts arranged in a hierarchy roughly having depth 14. Although large in size, GO is mostly a class hierarchy with only one property ("part-of"). The main limiting factor for our approach will be the number of tables and triggers a database system can realistically support. MySQL, for example, is limited only by the number of files possible on the operating system. Unless other DBMSs have strict limitations, we do not see scalability to be a problem in general since ontologies do not typically grow to sizes on the order of millions of concepts. To be clear, we mean scalability in terms of the conceptual model, not the data instances which definitely pose scalability issues. We tested our system on a toy ontology up to size 40,000 and depth 20 and there was no visible difficulty. In future work, we will process GO itself and possibly incorporate data instances from ZFIN and MGI given our strong working relationship with those groups.

The next goal for the NEMO project is a comprehensive ontology-based modeling and integration system that will facilitate the representation and dissemination of ERP data across different EEG and ERP analysis methods, different experiment paradigms, and different laboratories. It is likely that the representation of EEG and ERP patterns that are associated with different analysis methods and different functional (experiment) paradigms will require multiple ontologies to be developed. Ontology-based integration in NEMO will study the mapping rules between these EEG and ERP ontologies. Given the mapping rules between different ontologies, once the user query comes in, various ERP databases with different ontologies can be searched for answers to the query. We reported an efficient ontology-based data integration system called *OntoGrate* that addresses this problem using an inference engine [13]. In general, we anticipate that this research can be generalized for integrating other types of neuroscience data (e.g., event-related fields (ERF) and functional magnetic resonance imaging (fMRI) data) and can support other biomedical ontology-based data sharing efforts (e.g., GO) in the future. Figure 7 highlights the main idea behind the query answering scenario under this model of integration.

Acknowledgements

We thank the other members in the NEMO working project group, and in particular Robert Frank, Allen Malony and Don Tucker, for their collaboration on related work. We also thank Jeff Z. Pan and Zena M. Ariola for valuable discussions on theoretical aspects of this work.

References

1. MGI: Mouse Genome Informatics, http://www.informatics.jax.org/
2. Open Biomedical Ontologies (OBO), http://www.geneontology.org/GO.format.obo-1_2.shtml
3. Resource Description Framework, http://www.w3.org/RDF/
4. SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/
5. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/SWRL/
6. The National Center for Biomedical Ontology, http://www.bioontology.org/
7. Web Ontology Language (OWL), http://www.w3.org/TR/owl-ref/
8. ZFIN: The Zebrafish Information Network, http://www.zfin.org
9. Baader, F., Nutt, W.: Basic description logics. In: Description Logic Handbook, pp. 43–95 (2003)
10. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: International Semantic Web Conference, pp. 54–68 (2002)
11. Dien, J.: Addressing misallocation of variance in principal components analysis of event-related potentials. Brain Topography 11(1), 43–55 (1998)
12. Dou, D., Frishkoff, G., Rong, J., Frank, R., Malony, A., Tucker, D.: Development of NeuroElectroMagnetic Ontologies (NEMO): A Framework for Mining Brainwave Ontologies. In: Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining (KDD), pp. 270–279 (2007)
13. Dou, D., LePendu, P.: Ontology-based integration for relational databases. In: ACM Symposium on Applied Computing (SAC), pp. 461–466 (2006)
14. Fikes, R., Hayes, P.J., Horrocks, I.: Owl-ql - a language for deductive query answering on the semantic web. J. Web Sem. 2(1), 19–29 (2004)
15. Frishkoff, G.A.: Hemispheric differences in strong versus weak semantic priming: Evidence from event-related brain potentials. Brain Lang. 100(1) (2007)
16. Frishkoff, G.A., Frank, R.M., Rong, J., Dou, D., Dien, J., Halderman, L.K.: A Framework to Support Automated Classification and Labeling of Brain Electromagnetic Patterns. In: Computational Intelligence and Neuroscience (CIN), Special Issue, EEG/MEG Analysis and Signal Processing (2007)
17. Gardner, D., Knuth, K.H., Abato, M., Erde, S.M., White, T., DeBellis, R.: Common data model for neuroscience data and data model exchange. J. Am. Med. Inform. Assoc. 8(1), 17–33 (2001)
18. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for owl knowledge base systems. J. Web Sem. 3(2-3), 158–182 (2005)
19. Hull, R., King, R.: Semantic database modeling: survey, applications, and research issues. ACM Comput. Surv. 19(3), 201–260 (1987)
20. Keator, D.B., Gadde, S., Grethe, J.S., Taylor, D.V., Potkin, S.G.: A general xml schema and spm toolbox for storage of neuro-imaging results and anatomical labels. Neuroinformatics 4(2), 199–212 (2006)
21. Koslow, S.H., Subramaniam, S. (eds.): Databasing the Brain: From Data to Knowledge (Neuroinformatics). Wiley-Liss, Chichester (2005)
22. Laird, A.R., Lancaster, J.L., Fox, P.T.: Brainmap: The social evolution of a human brain mapping database. Neuroinformatics 3(1), 65–78 (2005)
23. Lindberg, D., Humphries, B., McCray, A.: The Unified Medical Language System. Methods of Information in Medicine 32(4), 281–291 (1993)

24. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between owl and relational databases. In: Proceedings of the 16th International Conference on World Wide Web (WWW), pp. 807–816 (2007)
25. G.Ontology Consortium. Creating the Gene Ontology Resource: Design and Implementation. Genome Research 11(8), 1425–1433 (2001)
26. Pan, Z., Heflin, J.: Dldb: Extending relational databases to support semantic web queries. In: Workshop on Practical and Scalable Semantic Systems (2003)
27. Paton, N.W., Díaz, O.: Active database systems. ACM Comput. Surv. 31(1), 63–103 (1999)
28. Reiter, R.: Deductive question-answering on relational data bases. In: Logic and Data Bases, pp. 149–177 (1977)
29. Reiter, R.: On closed world data bases. In: Logic and Data Bases, pp. 55–76 (1977)
30. Reiter, R.: On structuring a first order data base. In: Proceedings of the Canadian Society for Computational Studies of Intelligence, pp. 90–99 (1978)
31. Reiter, R.: What should a database know? J. Log. Program. 14(1&2), 127–153 (1992)
32. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice-Hall, Englewood Cliffs (2003)
33. Ullman, J.D.: Principles of Database and Knowledge-Base Systems, vol. I. Computer Science Press (1988)
34. Yu, A.C.: Methods in biomedical ontology. J. of Biomedical Informatics 39(3), 252–266 (2006)