MODELING CONTESTED CATEGORIZATION IN LINGUISTIC DATABASES

By

Jeff Good
Max Planck Institute for Evolutionary Anthropology, Leipzig
and
The Rosetta Project, San Francisco

and

Calvin Hendryx-Parker
Six Feet Up, Inc.

Paper presented at

2006 E-MELD Workshop on Digital Language Documentation
Lansing, Michigan
June 20–22, 2006

1 Introduction

A fundamental problem in the design of linguistic databases is finding effective ways to encode content which is of a contested nature—that is, content which involves data for which there is no general consensus on how it should be best interpreted.[1] A clear example of such content is the grouping of languages into genealogical units. Numerous proposals abound, but only a relatively limited subset of these proposals are not, in some way, contested. For example, while current wisdom accepts the existence of numerous low-level genealogical units (e.g., Germanic or Algonquian) and a number of high-level units (e.g., Indo-European or Niger-Congo), there is no general consensus on the grouping of the larger families together or, for the most part, on the arrangement of the subgroupings within even relatively small families. There are proposals, of course, some of which have more support than others, but there is no consensus.[2]

In the ideal world (from a technical perspective), none of the information we may want to put into a database would be contested in any way. However, it is often undesirable to exclude such information for both practical and analytical reasons. On the practical side, certain kinds of contested content may be very useful to the user of a database—even to a user who thinks the content is "wrong". Again, borrowing from the domain of genealogical relationships, navigating through information about the 7000+ languages of the world can be greatly facilitated by putting languages into genealogical groupings which allow traversal up and down a "family" tree to discover information about languages "related" to some other language. Since, at present, there is no non-contested family tree for all the world's languages, this sort of feature can only be implemented by including at least some contested content in a database.

On the analytical side, if we want a database to be useful not only to impart "textbook" information but also to play a role in working on "cutting-edge" research problems, it will often be necessary for it to include contested content so that different hypotheses embodied by that content can be usefully compared. For example, if there are two competing subgroupings of a given family, it could be quite valuable to encode both in a single database in a way which would allow a user to see how each subgrouping fares against different data sets. The database could then be used to establish that one of the proposed subgroupings is very likely to be correct, thus shifting its status from contested to non-contested—a clearly desirable outcome which is predicated on contested content being encoded in the database in the first place.

However, making use of non-contested and contested data in the same database raises a number of technical issues. One of these is ensuring that the overall structure of the database does not crucially depend on contested content because of the likelihood that such content will be quickly rendered obsolete. Another is that it is not clear how best to encode contradictory information embodied by contested hypotheses within a single database structure, especially since the contradictory portions of the hypotheses will typically only to apply to quite specific parts of the

[1] The first author of this paper was primarily responsible for its authorship. The second author was primarily responsible for implementation of the database system (with the help of other individuals working for Six Feet Up, Inc.). We would like to acknowledge Jim Mason, Gary Simons, and Michael Cysouw for their their advice on the work described here.

[2] Of course, it should go without saying that a "consensus view" should not be equated with the "correct view". It simply reflects what one community generally believes to be correct based on the evidence available to it at any given point in time.

database's overall content.

The present paper outlines one solution to the problem of encoding both non-contested and contested content in a single linguistic database, as conceptualized for and implemented in the Rosetta Project's All Language Database.[3] As will be seen, a solution first devised to deal with problems relating to the encoding of multiple cross-cutting—and possibly contradictory—classifications of languages, dialects, and families, has been usefully extended to many other parts of the database, which, while not including contested content, were usefully encoded with a similar data model as the contested data. In addition, in this particular case, Rosetta's technical requirements and its desire to create an interoperable, long-lived language database converged on the same solution, and the implementation to be discussed here conforms in important respects to some of the best practice recommendations for content interoperability and longevity as envisioned by the Semantic Web framework[4] and by initiatives like E-MELD.

While much of the content of the Rosetta database is distinct in important ways from the content of the resources produced by documentary and descriptive linguists, we believe that the general model for dealing with contested data described in this paper will be applicable to a wide range of language resources and linguistic data types. Therefore, even though the problem space covered in this paper does not coincide with the prototypical concerns of the ordinary working linguist, we believe that it may still contain a number of relevant lessons for designing tools and standards relevant to their work.

The structure of this paper is as follows: Section 2 gives a descriptive overview and model of the kinds of information found in the Rosetta database, highlighting the distinction between contested and non-contested content. Section 3 describes our current implementation of this model. Section 4 offers a brief conclusion.

2 Rosetta's data structures

2.1 Introduction

The Rosetta Project has the goal of assembling a database containing all information available on the languages, dialects, and language families of the world. Such an ambitious goal will probably never be fully reached, of course. Nevertheless, having such a goal in mind has forced us to seriously consider what kind of database structure can best facilitate such a massive undertaking. Of particular concern has been devising a data model which has enough structure to allow users to easily navigate through the information we have collected, while avoiding making our overall system unnecessarily dependent on "unstable" content.

This data that is most problematic for Rosetta, in this regard, involves genealogical relationships between languages and language families. Important aspects the site's functionality depend on having a complete genealogical classification of all the world's languages in the database. In particular, such information allows (i) users to navigate up and down a "family" tree in order to move between languages and families in a user-friendly and intuitive way and (ii) allows the site to aggregate information both up and down the family tree so that users looking at information about a particular language, dialect, or family can easily discover how much information is available on linguistic units closely related to it.

As is well-known to linguists, there is no universally-accepted genealogical classification of

[3] http://rosettaproject.org/

[4] http://www.w3.org/2001/sw/

the world's languages. In order to provide "family tree" functionality, we must, therefore, choose a classification that is not universally accepted. Over time, however, we expect (perhaps in part through use of Rosetta's database) parts of the classification will be improved and revised, and our database will need to be updated accordingly. This raises an immediate technical problem: How can we effectively use one classificatory system today but be able to easily replace it with a better system tomorrow?

While the problems of genealogical classification are particularly salient in this regard, the same basic issues are found in a number of other types of information in the database. For example, in addition to genealogical navigation, the site supports geographic classification. While geographic relations are less contested than genealogical relationships, at least in linguistic circles, they are not necessarily stable—especially as regards politically-defined geographic units. At least for the foreseeable future, it is unwise to assume, in the context of database design, that the current political partitions of the world will remain constant. We, therefore, need to encode at least some aspects of the geographic content of the database in ways which afford similar flexibility to what is needed for genealogical relationships.

Another problematic area of database content, worth mentioning in this context, is the basic classification of linguistic entities like languages, dialects, or families into one of those three types. It is a truism in linguistics that the line between a language and a dialect cannot be firmly drawn. And drawing such lines is not only relevant to determining whether or not some unit is a language, it is also central in determining if some unit a family. If the varieties of Chinese, for example, are considered dialects, then Chinese would be a language. If they're considered languages, then Chinese would be a family. Therefore, in addition to being flexible about genealogical and geographic relationships, the database cannot even crucially rely on the idea that some named linguistic unit is a "language" or a "family"—since these sorts of classifications, too, may change over time.

In the rest of this section, we will discuss the conceptual model we have developed for the content of the Rosetta database to help deal with issues regarding such types of contested data. In section 3, we will, then, be in a position to discuss how we have been able to implement this conceptual model. In the discussion to follow, for the sake of concreteness, we will focus on problems relating to genealogical classification—though most of the points to be raised will be relevant for the other problems just mentioned, and they will sometimes also be referred to.

Rosetta's conceptual model rests on making a fundamental distinction between nodes and relationships. The former are understood to be relatively stable, non-contested parts of the database, and the latter are used to encode contested information. Each is discussed in turn.

2.2 Nodes

In the Rosetta database architecture, nodes correspond to stable, non-contested content. At present, there are two primary types of nodes, which are given in (1).

(1) a. **Lingual Nodes:** Database elements corresponding to linguistic entities—languages, dialects, families, etc.

b. **Geographic Nodes:** Database elements corresponding to geographic entities (both purely geographic and political)

Critically, for Rosetta, for an entity to qualify as a lingual node or a geographic node, it is not important whether or not its existence as a "real" entity is non-contested or not. All that is required is for someone to have referred to that entity. So, for example, any entity ever given a Ethnologue code—in any version of the Ethnologue—could be treated as a node in the database. A node is, therefore, the analog of a *proposal* for a linguistic entity, not an analog of a linguistic entity itself. Whether or not an entity has been proposed will, generally, be non-contentious (one need only find a relevant reference). What is contentious is how that node relates to the real world.

At present, the Rosetta database only contains nodes corresponding to Ethnologue 14 languages and families and, in some cases, nodes for dialects of entities classified as languages in Ethnologue 14.[5] This has allowed us to avoid devising a solution to a problem that immediately arises from the idea that a node corresponds to a proposal and not to a real entity: How do we deal with proposals for units which are supposed to correspond to the "same" real-world entity, but might not be exactly the same. For example, is Greenberg's Niger-Congo the same thing as Ethnologue 14's Niger-Congo? While Rosetta has not yet had to implement a solution to this problem, we believe it can be accommodated in our current system, and we will come back to this issue in section 3.5.

A Rosetta node has three properties, given in (2).

(2) a. **Identifier:** Every node has a unique identifier.

b. **Metadata:** Every node is associated with a limited amount of metadata. This includes, for example, a human-readable name for the node, information about what kind of node it is (e.g., lingual vs. geographic), and an indication of what documents the node is associated with (e.g., digital documents or references to books, articles, etc.)

c. **Documents:** Each node can be associated with digital documents or references to books, articles, etc.

Figure 1 schematizes three Rosetta lingual nodes, for English, Russian, and Slavic. Each node is marked with an identifier and is associated with a metadata record and a set of documents. In addition, the metadata record points to the documents the node is associated with.[6]

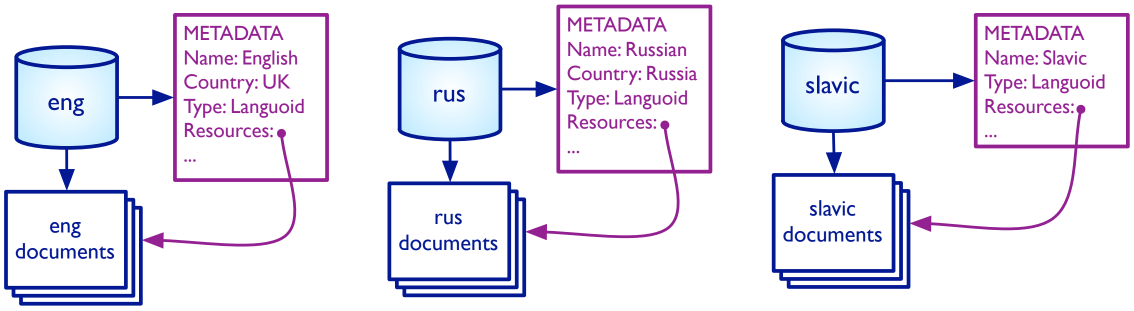

Figure 1: Nodes in the Rosetta database

[5] The reliance of Rosetta on Ethnologue 14 is purely historical in nature. In the near future, we hope to move to Ethnologue 15.

[6] Strictly speaking, this information is not required for the system to be operative. However, modeling the structure this way has been proven convenient in the overall operation of our database system.

As indicated in figure 1, a node can be understood as a kind of container, in which documents can be placed. The metadata associated with gives some basic information about what kind of container the node is and what it contains.

Critically, all nodes are conceptualized as belonging flat structure, separate from any information indicating how they may be grouped genealogically or geographically—or in any other way except for very high-level groupings like lingual versus geographic. The modeling of structural relationships among nodes is the topic of the next section.

2.3 Relationships

Content which is considered to be non-stable and potentially contested in the Rosetta database is modeled as relationships among nodes and between nodes and other data types. Such relationships can be schematized as in figure 2.

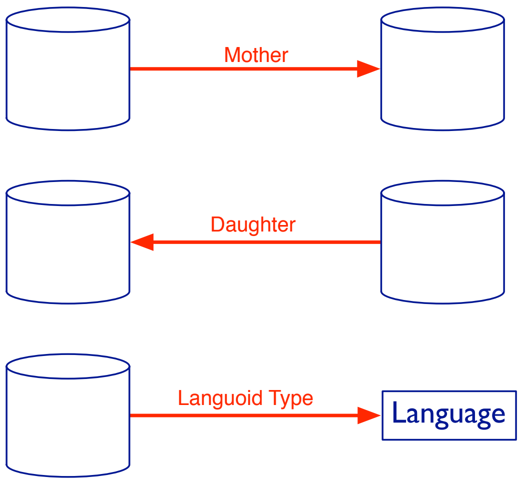

Figure 2: Relationships in the Rosetta database

Three possible kinds of relationships are given in figure 2: a relationship where one node is the genealogical mother of another node, a relationship where one node is genealogical daughter of another node, and a relationship where a lingual node is classified as a language (as opposed to a family or a dialect).[7]

As implied in figure 2, there are two broad classes of relationships in the Rosetta database, *nodal* relationships and *classificational* relationships. These are defined in (3).[8]

[7] Figure 2 makes use of the term *languoid* which Rosetta uses, at present, as a cover term for any type of lingual entity: language, dialect, family, language area, etc. It is roughly similar to the term *taxon* from biological taxonomy, except it is agnostic as to whether the relevant linguistic grouping is considered to be genealogical or areal (or based one some other possible criteria for grouping languages). Another word which as been suggested (by David Gil) for this concept is *lange* [lænǰ]. Further suggestions/advice on how to name this concept are actively solicited.

[8] From an implementation standpoint, in fact, there are only nodal relationships in the database, contrary to what is implied in (3). Even things which, at least from the linguist's perspective, are not obviously "nodal", like the classifier *language* or a language code are treated as a nodes—nodes of different types than the lingual or geographic nodes mentioned in (1). We distinguish between nodal and classificational relationships, conceptually, here, since we believe this reflects a "folk" classification of most linguists that these kinds of content are distinct, with the former being closer to real "data" and the latter closer to metadata.

(3) a. **Nodal relationship:** A relationship holding between two nodes in the database. Prototypical cases includes mother/daughter relationships between nodes and relationships holding between lingual nodes and geographic nodes (e.g., that a language *X* is spoken in country *Y*).

b. **Classificational relationship:** A relationship stating that a node is of some type or has some attribute. Prototypical cases include specifying whether or not a lingual node is classified as a language, dialect, or family or specifying that a given node is associated with a language code for a group other than Rosetta.

Clearly, some of the relationships in the Rosetta database will be more contested than others. Many of the genealogical relationships are likely to be highly contested, while some of the classificational relationships are likely to be relatively non-contested. For example, many of the classificational relationships in the database amount to statements like: "Rosetta lingual node *X* corresponds to Ethnologue entity with code *YYY*." The reason for treating even this kind of information as a relationship, rather than incorporating it directly into nodal metadata, is not that, in and of itself, it is contested or unstable. Rather, its utility is unstable. Another organization's language classification may be quite useful at a particular point in time but less useful in the future. We, therefore, treat such information in the same way as truly contested content to ensure our database structure is not overly reliant on it.

2.4 Combining nodes and relationships

To this point, the innovative nature of the Rosetta database—at least for linguistic work—may not be completely clear. Clearly, other databases have had node-like elements and relationship-like statements. The online version of the Ethnologue, for example, makes use of such a database. What is different about Rosetta's database structure (at least, we think it is) is that nodes and relationships are completely partitioned from each other and treated as independent data sets. Their only point of interconnection is through nodal identifiers, which relationship statements can refer to. To understand the nature of this partition of the data sets, it is useful to compare figure 3 and figure 4.

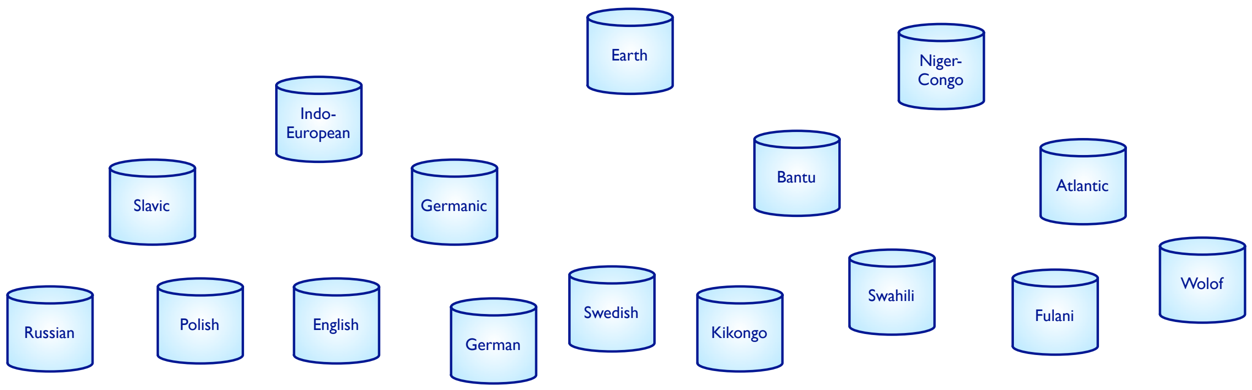

Figure 3: Subset of unstructured nodes from the Rosetta database

Figure 3 schematizes a set of lingual nodes as an unstructured set of containers, holding information about lingual entities. All the nodes in figure 3, except one, refer to languages or families.

The one exception is the node *Earth* which is used in the database as a node corresponding to the set of all lingual entities spoken on Earth.

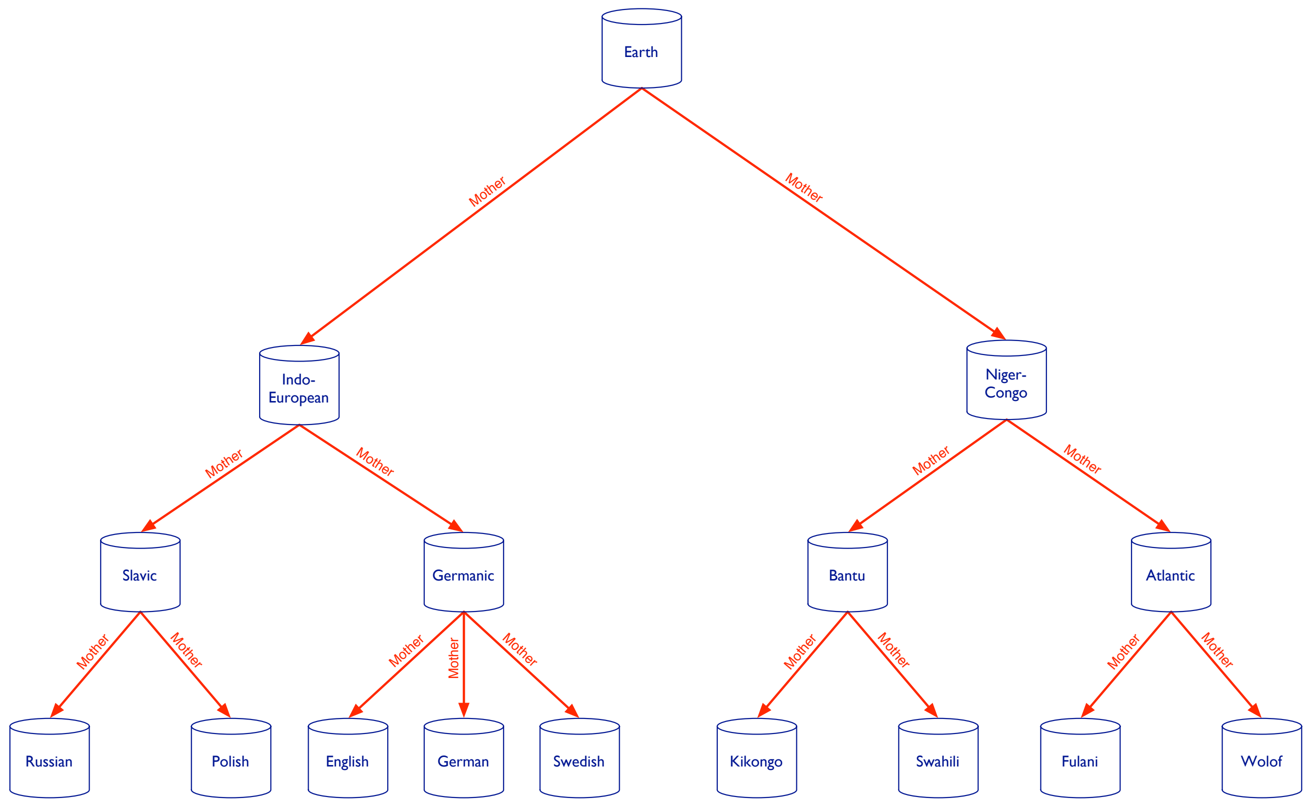

Figure 4: Subset of relationships from the Rosetta database

Figure 4 schematizes a set of (genealogical) mother-daughter relationships holding among the lingual nodes depicted in figure 3. The "empty" containers in figure 3 are intended to represent the fact that the relationships are not stated directly with respect to the nodes but, rather, only refer to the nodes.

In order to make use both of information included as part of lingual nodes and of information encoded as relationships, the nodal data structures and the relationships must be merged. Such a merger is schematized in figure 5, where "full" containers replace the empty, placeholder containers given in figure 4. Crucially, this data merger is done in response to particular user requests for information from the database and is created in "real-time". That is, the data is never stored in merged form but, rather, is only merged temporarily when such merger will facilitate specific tasks. The temporary nature of this process of database merger is analogous to the creation of a temporary working format or display format for a lexicon or a set of texts.

Partitioning the database into a set of nodes and a set of relationships requires Rosetta to merge the two data sets in order to achieve many basic types of desirable functionality (e.g., using a family tree to discover documents associated with related languages). The partition, therefore, puts extra burdens on Rosetta's database tools since they need to both retrieve *and* merge information from the database. However, we believe these burdens are outweighed by the benefits of such a system which gives our database two crucial features: flexibility and longevity, as summarized in (4).

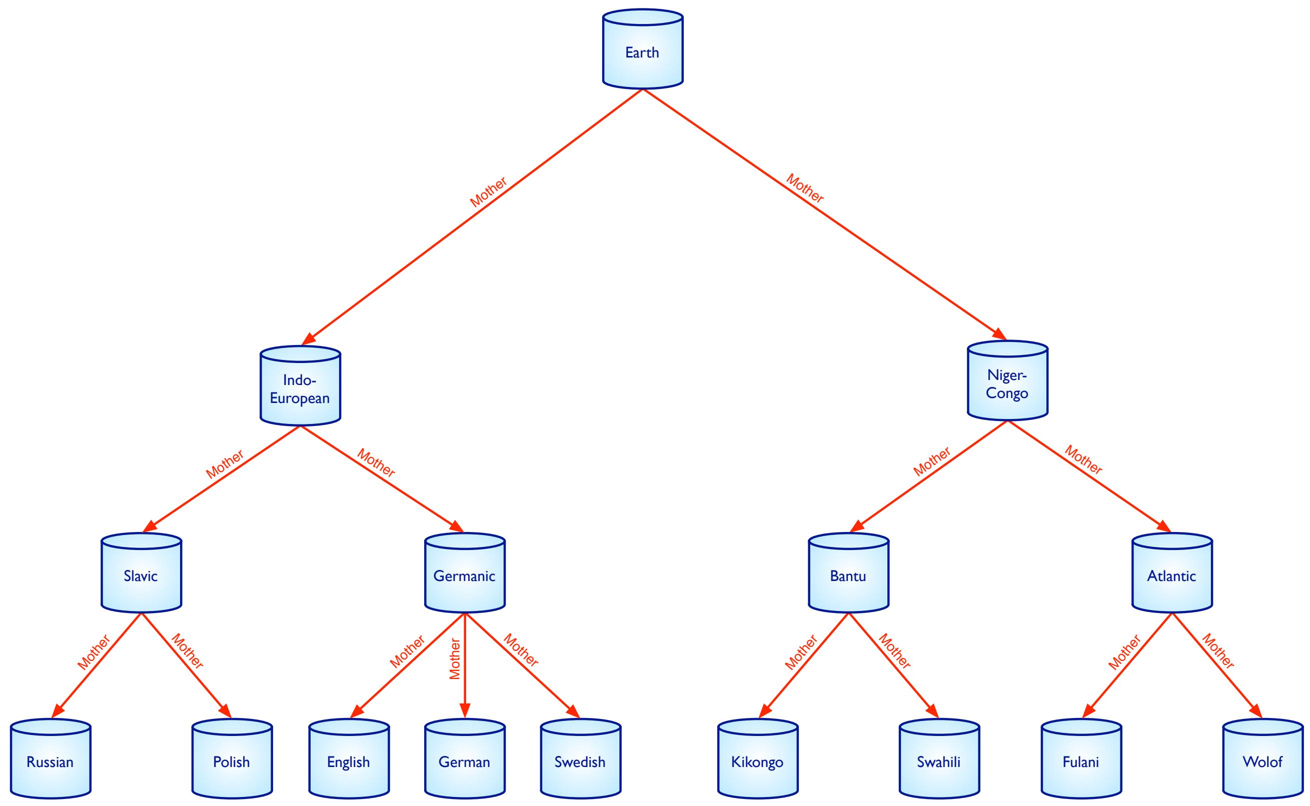

Figure 5: Merger of nodes and relationships into a unified structure

(4) a. **Flexibility:** The database's structure does not refer, in any crucial way, to any classificational structure for lingual nodes. Therefore, different sets of nodal relationships can be merged with the nodes, as appropriate (assuming those relationships have been coded in the database).

b. **Longevity:** Nodal content, which is expected to be relatively stable, is not dependent on relationship content in any crucial way, thereby ensuring that changing relationship content will not adversely affect the ability of the database to encode and store nodal content.

To give a concrete example of the benefits of the flexibility of the Rosetta database structure, consider figure 6 which gives a different organization for the language nodes seen in (3). In figure 6, the nodes are organized according to geographic relationships—specifically what continent they are found in.[9]

Geographic organization is only one other possible way of organizing lingual nodes. The database architecture, described here, could also be used to create any number of arbitrary nodal organizational structures, assuming the relevant relationships were stored in the database. One could imagine, for example, organizing nodes based on linguists who work on them. One could

[9] Geographic entities, of course, can refer both to areas defined primarily based on linguistic criteria and areas defined based on other criteria, e.g., topographic ones. The current form of the Rosetta database, in principle, would allow for either a "linguistic" geographic classification or a geographic classification of any other desired type.

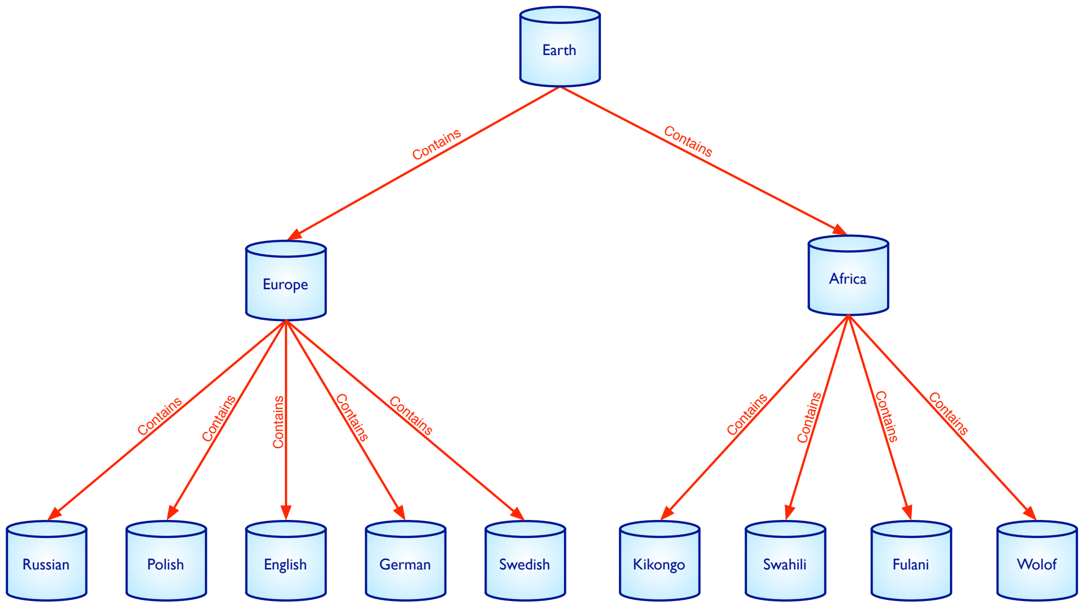

Figure 6: A geographically-based merger of nodes and relationships

also imagine overlaying two separate organizational systems on top of each other to see how they compare. Achieving such functionality would require the development of techniques and tools beyond what is currently provided, but, crucially, unlike a database which coded relationships among nodes directly into a nodal storage system, the Rosetta database itself provides no barrier to the development of such services.

We will return to the issue of database longevity in section 3.4 where we describe a straightforward pathway through which the content associated with nodes and content associated with relationships can be transformed into archival formats storing both contested and non-contested information.

In the next section, we will give an overview of how the database system just described has been implemented. The core of the implementation is to create, effectively, two database partitions, one containing (stable) nodal information, following the schema in figure 3 and the other containing (unstable) relationship information, following the schema in figure 4.

3 Implementation

3.1 Introduction

In this section, we will discuss how the database system described in section 2 has been implemented. While not all aspects of the implementation will be relevant to other projects, our management of relationships among nodes, we believe may be applicable to a wide variety of problems involving the organization of linguistic data and, therefore, should be of interest to the E-MELD community.

3.2 Nodes as objects

The Rosetta Archive and web site are built on the open-source Plone content management system[10], which is itself built on the open-source Zope application server[11]. Zope is written in Python[12], an open-source programming language which has been adopted by a number of other linguistic tool building projects, most notably the Annotation Graph Toolkit[13]. As a result of the fact that Rosetta is already making use of the Zope application server, we have also chosen to use the Zope Object Database (ZODB)[14] for some of our nodal storage needs.

As implied by its name, a ZODB is an object-oriented database. Such a database can have largely similar functionality to a relational database. However, rather than store its data in the form of tables, it stores it in the form of objects. These objects can be assigned properties, as needed, which correspond, roughly, to relational database fields. We could thus model the ZODB representation of a Rosetta lingual node schematically as in (5).

(5)
```
NODE OBJECT
Properties
  ID: rus
  Metadata:
    Name:  Russian
    Type:  Lingual
    Documents:  RussianText1, RussianText2, RussianGrammar1
```

The representation in (5) gives the ID and properties of a lingual node for Russian. The properties include a human-readable name for the node, an indication that this is a lingual node, and a list of references to documents associated with node.[15]

Depending on the nature of the documents, they may be included directly in the ZODB, becoming, in effect, another property of the node. Or, if they are of a type which is impractical for storage in the ZODB for various technical reasons (e.g., large sets of scanned image files), the node object will contain information necessary for successful external access of those documents, typically in the form of a URL pointing to a Rosetta server where those materials are stored.

The ZODB's object-oriented database model gives us a natural way to implement our model of Rosetta nodes, since it conceptualizes its basic stored elements as objects, just as we conceptualize our nodes as objects. However, Rosetta nodes themselves are relatively simple data structures, and it would not be particularly difficult to also encode the information found in them in a relational database. The only truly crucial feature a node must have in order to interface with a database of relationships, in the present model, is a unique identifier.[16]

Section 3.4 will discuss how the content of Rosetta nodes can be expressed in an archival format.

[10] http://plone.org
[11] http://zope.org
[12] http://python.org/
[13] http://agtk.sourceforge.net/
[14] http://www.zope.org/Wikis/ZODB/
[15] As will be discussed in section 3.3, the Rosetta database, at present, has an RDF component. Some of the nodal content discussed here is actually stored as RDF associated with a node but not stored directly with the nodal object.
[16] The unique identifiers used by Rosetta are more complex than the one given in (5). As will be seen in section 3.3, Rosetta actually uses URI's for such identifiers.

3.3 Relationships as RDF triples

Unlike the nodal portion of the database, the primary storage format for the relationship portion of the database is not tied to a specific platform but is, rather, expressed using the formal mechanisms of Resource Description Format (RDF)[17], one of the component technologies currently underlying the Semantic Web[18]. One of the products of the E-MELD initiative, the GOLD ontology[19], is also grounded in the Semantic Web, and the use of RDF in order to express information about linguistic data has already been raised in the context of E-MELD. Therefore, while it is not clear if many other projects would be able to usefully adopt our implementation for storing the content of nodes, our implementation for the storage of relationships is likely to be of wide interest within the E-MELD community.[20] Simons' (2005) contribution to E-MELD[21] contains a useful introduction to RDF in the context of research on the digital expression of linguistic data. I will not give a complete description of RDF here and refer the reader to Simons (2005) for further discussion.[22]

The core principal of RDF is that a semantic representation of the content of some piece of data can be expressed in the form of a special class of statements referred to as *triples*, which consist of a *subject*, a *predicate*, and an *object*. A given triple expresses that the relationship referred to by its predicate holds between its subject and its object. Triples can be expressed in a number of ways, with the most intuitive being in the form of a (visual) graph. A simple such graph, expressing a single triple drawn from the Rosetta relationship database is given in figure 7.[23]

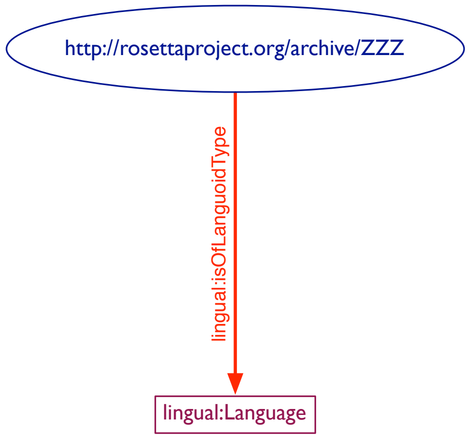

Figure 7: An RDF triple from the Rosetta relationship database

[17] http://www.w3.org/RDF/

[18] http://www.w3.org/2001/sw/

[19] http://www.linguistics-ontology.org/

[20] The discussion in this section by no means exhausts the features and capabilities of the Rosetta RDF system, which cannot be fully enumerated here. Two capabilities worth specifically mentioning which will not be discussed below are (i) its support of a full text search via a mechanism that builds a separate graph on top of the primary data graph pointing to all the space-delimited strings used within that graph and (ii) support for SPARQL (see http://www.w3.org/TR/rdf-sparql-query/) a powerful RDF query language. These features are built in to RDFlib. The former was partially developed for RDFlib in the context of Rosetta's work on its RDF database.

[21] http://emeld.org/workshop/2005/papers/simons-paper.pdf

[22] For a non-linguistic, technical introduction to RDF, see also: http://www.w3.org/TR/rdf-primer/.

[23] Following W3C convention, URI's in the RDF graphs in this paper will be represented by ovals and string literals by rectangles.

The graph in figure 7 states that an entity with URI http://rosettaproject.org/archive/ZZZ is the subject of a predicate *lingual:isOfLanguoidType* which has the object *lingual:Language*. The entity referred to by the subject URI is the language which has Ethnologue 14 code ZZZ (Dimli, an Iranian language).[24] This URI serves as the unique ID which allows nodal content and relationships to be merged along the lines of what was discussed in section 2.4. The prefix *lingual:* marking the predicate and the object refers to the XML namespace[25] in which the terms *isOfLanguoidType* and *Language* are defined.[26] The term *languoid* in the predicate *isOfLanguoidType* is a cover term employed by the Rosetta RDF system for a dialect, language, family, or other kind of linguistic grouping. This triple, therefore, expresses the (rather uninteresting) claim that would be rendered in English as "Dimli is a language".

Much of the power of RDF comes from the fact that triples can be joined to form networks of triples. Figure 8 exemplifies this by adding a number of other triples to the graph in figure 7.

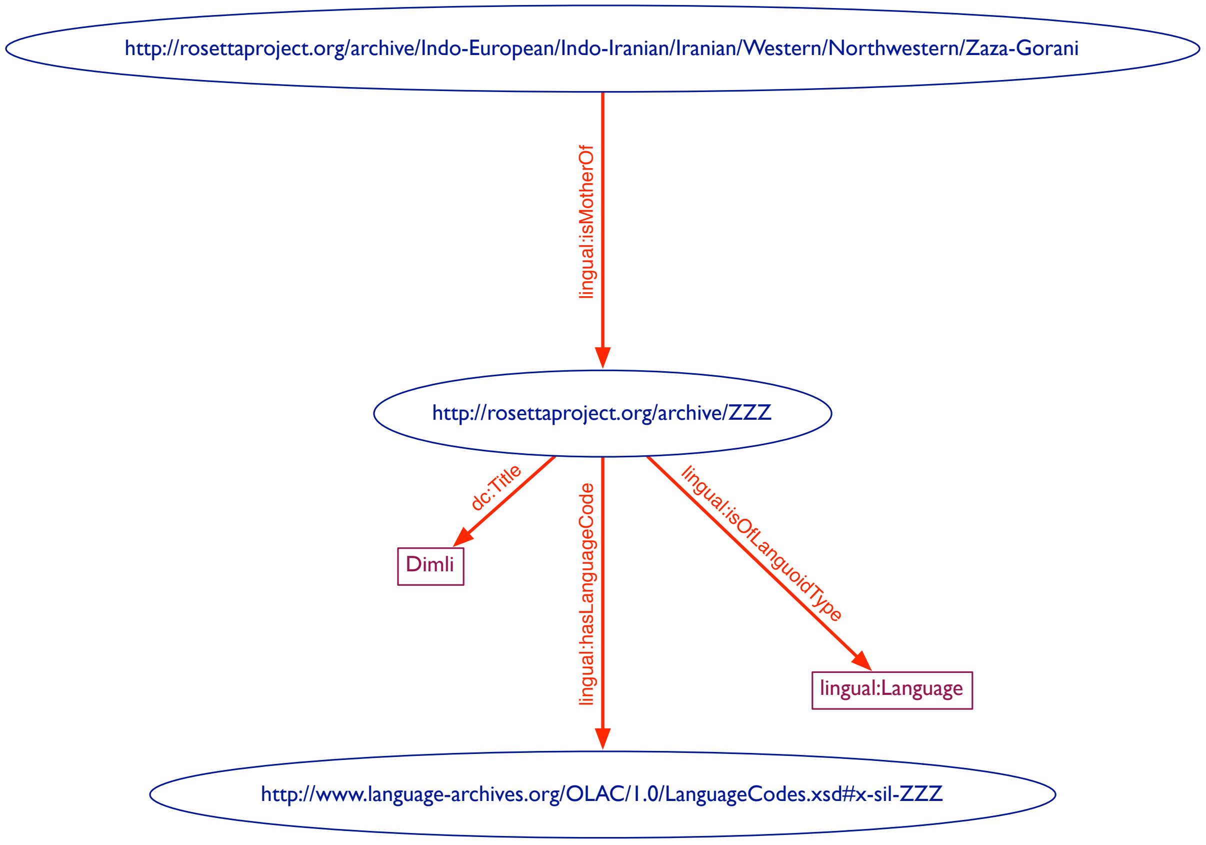
http://rosettaproject.org/archive/Indo-European/Indo-Iranian/Iranian/Western/Northwestern/Zaza-Gorani
lingual:isMotherOf
http://rosettaproject.org/archive/ZZZ
dc:Title
Dimli
lingual:hasLanguageCode
lingual:isOfLanguoidType
lingual:Language
http://www.language-archives.org/OLAC/1.0/LanguageCodes.xsd#x-sil-ZZZ

Figure 8: A small network of RDF triples from the Rosetta relationship database

The set of triples in figure 8 can be expressed in English as follows: A lingual node in the

[24] While Rosetta uses Ethnologue 14 codes in many of the URI's of its archives. It should not be assumed that Rosetta will maintain any isomorphism between codes indicated in its URI's and any Ethnologue coding scheme. Language codes associated with lingual entities are coded by the use of RDF triples explicitly linking a code to the entity URI as in figure 8.
[25] For an introduction to XML namespaces, see: http://emeld.org/school/classroom/xml/howto.html.
[26] This is namespace maintained by Rosetta. The URI for the namespace is: http://rosettaproject.org/lingual/.

Rosetta database which is commonly known as Dimli is a language with OLAC language code x-sil-ZZZ, and it is a daughter of the Zaza-Gorani language group.[27] Note that one of the predicates for the triples in figure 8 does not make use of the lingual namespace but, rather, makes use of the "dc" (i.e., Dublin Core[28]) namespace. (Rosetta uses the Dublin Core Title element as a predicate for expressing a relationship between a lingual node and a human-readable name for the node.)

As a standard of the Worldwide Web Consortium (W3C)[29], RDF is associated with a specification for how to express RDF triples in XML, along with other less verbose machine-readable formats.[30] This XML representation is simply a "serialized" version of the content of the RDF—that is, the two-dimensional graph is translated into a form which can be expressed as linearized XML without any information being lost. The primary format for Rosetta's database of relationships is this RDF/XML format. At present, the database contains statements of the relationships holding among languages and language families given in Ethnologue 14, statements of relationships between language areas and countries and continents, and statements relating languoids and geographic areas to various external coding schemes.

As XML is a poor format for fast processing, the Rosetta web site makes use of a higher-speed RDF database backend for handling actual user requests. At present, the backend is a ZODB database, of the sort described in section 3.2. However, the relationship data is stored separately from the nodal data and another database system (e.g., a SQL database) could also serve as the high-speed backend for the RDF database with minimal changes to the site.[31] Interaction with the RDF graph is mediated by a Python library, custom built for the Rosetta database on top of a general RDF library written in Python (RDFLib)[32].

While the driving force behind the use of RDF in the Rosetta site was to construct a database model where unstable, contested relationships could be stored, we have found RDF to also be a useful mechanism to express certain kinds of non-contested content on the site. For example, an important class of content on the site are references to books and articles on various languages. Given that the Zope application server incorporates Dublin Core directly into its metadata architecture and Dublin Core also has an RDF specification[33], we found Dublin Core RDF, augmented with RDF specifications of the OLAC extensions[34] to Dublin Core, to be a sensible choice for the storage of references in the database. Furthermore, in the near future, as we develop user and

[27] The URI http://rosettaproject.org/archive/Indo-European/Indo-Iranian/Iranian/Western/Northwestern/Zaza-Gorani appears to be a representation of a structured family tree path. This is, in fact, merely a mnemonic device and the sequence "Indo-European/Indo-Iranian/Iranian/Western/Northwestern/Zaza-Gorani" is interpreted as a unique identifier within the context of the archive. This convention was adopted for language families since Rosetta's current content is based on the Ethnologue 14 families and, unlike language codes, there is no guarantee that a given Ethnologue group name (e.g., "Western") will be unique. However, the entire family "path" should be unique. So, it is used as the unique identifier in the archive. Since the three-letter Ethnologue language codes are unique, the unique identifier for Ethnologue 14 languages in the context of the archive is simply the language code, hence the asymmetry in the family URI and the language URI in figure 8.

[28] http://dublincore.org/

[29] http://www.w3.org/

[30] A specification for RDF/XML can be found at: http://www.w3.org/TR/rdf-syntax-grammar/.

[31] See: http://www-db.stanford.edu/˜melnik/rdf/db.html for discussion of possible ways to store RDF in relational databases. Other backends supported by our RDF system at present include Sleepycat and MySQL.

[32] This latter library can be found at http://rdflib.net/.

[33] http://www.dublincore.org/documents/dcmes-xml/

[34] http://www.language-archives.org/REC/olac-extensions.html

community tools on the Rosetta site, we plan to use friend-of-a-friend RDF[35] for expressing information about individuals and how they relate to each other. Finally, some of the nodal content, discussed in section 2.2 and section 3.2 is also stored in RDF.

However, even though we have employed RDF, in some cases, to encode such non-contested content, the database still maintains a clear partition between contested and non-contested information through the use of RDF *contexts*.[36] RDF contexts allow us, effectively, to assign metadata to sets of triples indicating that they belong to particular classes, and we assign contested content to different classes from non-contested content. At present, we have a *default* context, where non-contested nodal metadata is stored. We also make use of three other contexts, one for encoding genealogical relationships, one for encoding geographic relationships, and one for encoding document references (including information on what languoids are the subject of those documents). The first two contexts contain the primary contested content found in the database at present. The last context, like the default context, contains non-contested content.

The RDF model for the Rosetta database is, at present, described in two RDF schemas, which are still partially under development. One of these schemas, the lingual schema, specifies properties of languoids, relationships among languoids, and relationships among languoids and geographic entities.[37] The other schema, the lingbib schema, which is much simpler, is used for references to language documents (books, articles, etc.). It extends the Dublin Core RDF Schema with OLAC's Dublin Core metadata extensions.[38] In addition, we are experimenting with linking the geographic categories specified in the lingual schema to those found in the SWEET Earth Realm ontology[39].

3.4 Comments on archiving and interoperability

Having described the implementation of the Rosetta database, it would be useful to comment briefly here on issues relating to archiving and interoperability.

All portions of the database stored in RDF/XML already make use of an archive-ready, interoperable format as their primary storage format—this is a tremendously valuable feature of RDF/XML. As discussed above, for reasons relating to performance, the Rosetta web site uses a ZODB working format of the relationships database. RDF triples can be added and removed to this working format. However, this working format still represents the content of the database as RDF, even though it doesn't use RDF/XML. The process of going from the ZODB working format to RDF/XML is trivial since such functionality is built-in the Python RDF library we use. Therefore, the route from working format to archival format presents no significant hurdles.

This leaves open the issue of how the content of Rosetta nodes can be archived and made interoperable. Some of the Rosetta nodal content consists of images and audio files. Such documents present the same archival and interoperability issues as they would in any digital archive. Since these are not problems specific to the Rosetta database, we set them aside here. However, the other content found on nodes, in particular the metadata associated with them, does present database-specific archival problems. Fortunately, the architecture we have built to maintain the RDF databases on the site can also be applied to this problem. While the primary form of the

[35] http://www.foaf-project.org/

[36] See: http://www.ninebynine.org/RDFNotes/UsingContextsWithRDF.html.

[37] The current version can be found at: http://emeld.org/workshop/2006/papers/goodparker-rdf/lingual.rdfs.

[38] The current version can be found at: http://emeld.org/workshop/2006/papers/goodparker-rdf/lingbib.rdfs.

[39] http://sweet.jpl.nasa.gov/ontology/earthrealm.owl

nodal content database is not stored in RDF, the RDF tools used by Rosetta can straightforwardly translate nodal objects in the ZODB into RDF, once a human user develops a translation from the object's properties to an RDF representation of those properties. Thus, nodal data can be straightforwardly transformed into an archival and interoperable format, not presenting any particular problems with respect to longevity or interoperability.

3.5 Outstanding issues

3.5.1 Introduction

There are two outstanding issues with respect to the RDF model for relationships described here, which we have not yet implemented a solutions for—because this has not been necessary—but which we expect to also be straightforwardly handled within an RDF database once this is required. We describe each in turn.

3.5.2 Reification

The first issue centers around the fact that, for many reasons, it may often be desirable to not simply encode a triple in the RDF database but to also annotate the triple itself. For example, we might want to indicate who was responsible for the creation of the triple. More substantively, we might want to say what the justification is for putting a triple in the database in the first place. Again, borrowing from the realm of genealogical relationships, it might be useful to say what the evidence is for a particular genealogical grouping. This situation is schematized in figure 9. Where the "justification" for claiming a genealogical relationship between Latin and several Romance languages is schematized as being the content of documents which are pointing towards the *Mother* predicates in the schema.

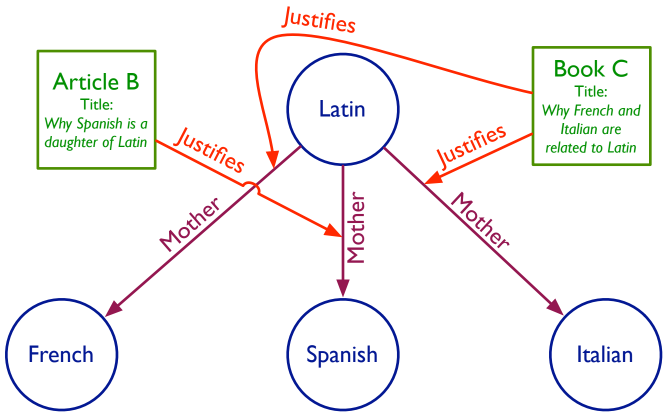

Figure 9: Annotating a triple

Annotating triples for information about their justification, as schematized in figure 9, could be extraordinarily useful for research since, in principle, it could allow a user to easily explore competing hypotheses with respect to some data set. In the context of Rosetta, it would allow us, for example, to include contradictory proposals regarding the genealogical classification within the database, annotating each proposed relationship for what sort of evidence was used to justify it. Users could then choose to extract proposed relationships based on different kinds of evidence in order to compare them—and, perhaps, ultimately, propose worthwhile refinements to existing claims.

The data in the Rosetta relationship database has been, to this point, limited enough in nature to allow us to skirt the issue of annotating triples. However, fortunately, when we reach a point when it will be important to do so, we will be able to exploit an existing RDF mechanism designed explicitly for this kind of problem. This mechanism is referred to as *reification*, and it effectively allows triples to be treated as subjects or objects in a graph which can then be components of other triples. Abstracting from certain technical details, reification can be schematized as in figure 10. The example in 10 is based on "Latin is the mother of Spanish" triple schematized in figure 9.

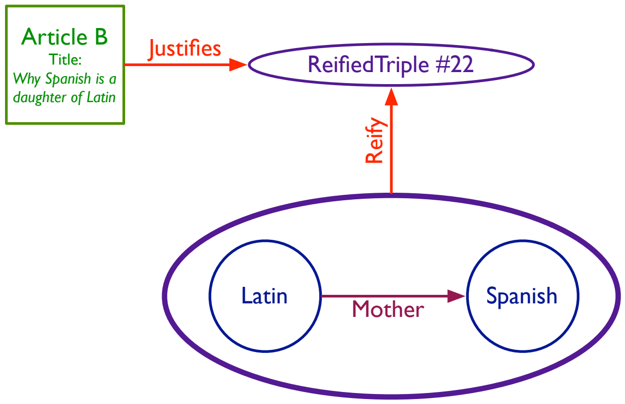

Figure 10: Reification of a triple

As schematized in figure 10, reification allows a triple to be treated as an entity in the graph, instead of as a statement. This entity can then be annotated by being treated as either the subject or object of another statement. In figure 10, the reified triple is treated as the object of the *Justifies* predicate.

Finally, it is worth pointing out that, while the Rosetta database has not yet made direct use of reification, it has used in indirectly when making use of contexts, discussed in section 3.3. This is because assigning triples to contextual sets amounts to reifying the triples and annotating those reified triples with other triples asserting they belong to the relevant set. We expect that the ability of our tools to manage contexts could be straightforwardly extended to add justification and provenance metadata of the sort schematized in figure 9.

3.5.3 One entity or two?

The second outstanding problem in the Rosetta database, which was mentioned above in section 2.2, is dealing with the issue of whether or not two linguistic entities generally thought of as the "same" are really the same. For example, is Greenberg's Niger-Congo the same thing as the contemporary notion of Niger-Congo? Conventional wisdom of the classification of languages into Niger-Congo has changed considerably since Greenberg's proposals. However, most linguists understand contemporary Niger-Congo to be essentially the same as Greenberg's Niger-Congo, but simply revised in ways which reflect the accumulation of new knowledge.

What concerns us here is not the problem of whether or not Niger-Congo—or any comparable "entity"—should be treated as one node or two nodes in the database. We take this to be an issue that can only be resolved by debate among linguists, and it is not the responsibility of the database to make a determination either way. However, we *would* like a mechanism which would allow any reasonable analysis of such cases to be encoded, including the possibility that it might be desirable to encode competing such analyses in the database. In the case of Niger-Congo example given above, this could mean including *three* nodes in the database: One for Greenberg's Niger-Congo, one for contemporary Niger-Congo, and one for "general" Niger-Congo where Greenberg's Niger-Congo and contemporary Niger-Congo are assumed to be one and the same entity

The structure of the Rosetta database, as described here, of course, would allow us to enter three such nodes into the database. The problem is making sure that, when this might happen, the database also contains information saying that these nodes could, under certain circumstances, sensibly be understood as all referring to a single entity. That is, we want to be able encode a situation like the one in figure 11, where three distinct nodes in the database are considered to be the same from the point of view of the (public) Rosetta web site.

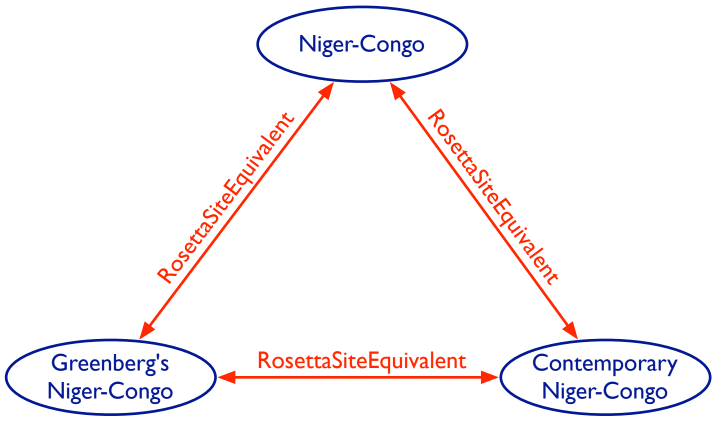

Figure 11: Equivalence among distinct nodes

Figure 11 schematizes the three Niger-Congo nodes as being equivalent in one particular context (the Rosetta Site). Elsewhere, they could still be considered distinct.

It should be clear that the sort of relationships schematized in figure 11 could be expressed via RDF triples, in the same manner as, for example, genealogical relationships. Therefore, while we have not yet had to implement a solution for the problem of not being certain when two invocations of the same ostensible linguistic entity really do refer to the "same" thing, we believe that such a case could be handled in the RDF database simply by making use of predicates specifically devised to encode different kinds of equivalence among nodes which are neither clearly the same or clearly distinct.[40]

Of course, implementing such multiple-node structures in this way will add a layer of complexity to the process of retrieving content from the RDF database. Nevertheless, for certain kinds

[40] The inspiration for dealing with this problem through a mechanism of equivalence localized to specific contexts comes from the use of *canonical equivalence* and *compatibility equivalence* within the Unicode standard (see http://www.unicode.org/reports/tr15/).

of data, this complexity may be a small price to pay for being to able to accurately the encode the lack of real-world consensus on whether or not a linguistic "entity" really is a single entity.

3.6 RDF databases and relational databases

It will not be possible here to give a full comparison of RDF databases to relational databases. However, since relational databases tend to be the tools of choice for complex databases in linguistic work, we will briefly discuss some of the advantages we have found so far to using an RDF system rather than a relational one.[41] We do this not to argue that RDF databases are, in general, superior to relational databases. Rather, we merely intend to point out some of the advantages they offer for modeling the data of interest at Rosetta.

First and foremost, RDF databases afford a level of flexibility of data encoding and extraction that, while not impossible, is much more difficult to achieve with relational databases. The reason for this is quite straightforward: While many data structures are well-represented as tables, there are a vast number of structures which are not. Many of the structures which Rosetta is interested are trees—in particular, trees which give cross-cutting classifications of the Rosetta nodes. The two most obvious examples of this are the genealogical tree, as schematized in figure 5, and the geographic tree, as schematized in figure 6.

If there were a single, non-contested tree associated with our nodes, then the table-based representations of relational databases would not be particularly problematic. We would use a series of tables to represent the one tree we were interested in modeling and query our data through the mediation of that tree. However, there is no such tree on which we can structure the database, and the overhead involved in storing multiple trees in a relational database, extracting just the tree we need for a given query, and then devising a way to make sure that tree is properly associated with nodal data becomes quite high. What is needed at Rosetta is a database system well-designed for storing, manipulating, and retrieving trees independently from other kinds of data. Since trees are a special type of graph, an RDF database was extremely well-suited to our needs.

We should stress here that, while the data area of most interest to us here was groupings of languages, this same basic problem with relational databases would apply to any data structure potentially involving competing trees—or, more generally, competing graphs. As discussed by Bird and Liberman (2001)[42], there are a range of cases where the analysis of primary linguistic data is also well-modeled with graphs. To the extent that some of this data would be usefully analyzed using multiple overlaying graphs, it, too, would seem a good candidate for being encoded as RDF.

A further advantage to RDF databases, in the context of Rosetta, is that they offer a degree of flexibility for reanalyzing and extending a given data set beyond what can be done with tabular representations of data. To take a hypothetical example, consider the typical workflow of a typological study: primary data samples are collected from available sources, and the primary data is used to classify languages into different types (e.g., SVO, SOV, VSO, etc.). For a one-time study, a table would be quite adequate for storing the relevant data: one column for a language name, one column for the primary data source/description, and one column for a typological class.

[41] As discussed in section 3.3, it is possible to use relational database tools as platforms on which RDF databases are built. Our concern here is with the structure of the databases from the developer's perspective, not its low-level implementation.

[42] Steven Bird and Mark Liberman. 2001. A formal framework for linguistic annotation, *Speech Communication* 33:23–60.

However, what if, a year later one wants to reanalyze the data using a more fine-grained classification than what was given originally? In principle, this, too, could be modeled with the table by adding a column with the new classificatory system. But, if both classificatory systems—the fine-grained one and the coarse-grained one—are useful ways of looking at the data, one might want both a new column *and* a further statement about how the two systems relate to each other. Perhaps, this could encoded using relationally using a separate mapping table. Now, suppose a year later a different linguist wants to look at the same primary data and perform a number of different kinds of classifications, some of which have sensible mappings to the earlier classifications. New columns and new tables are needed. Furthermore, these new columns and tables need to be marked as having been produced by a different researcher. So, now we need a way of encoding a new kind metadata into the system. Later, suppose a different linguist wants to add a completely different primary data set for the same set of languages and, then, introduce a set of typological classifications based on that data. Some parts of the original tables are shared, some parts are different. The original table would then need to be decomposed into two tables for the data to be properly encoded.

It is clear that this *could* all be done in a relational database. In fact, something like it has been done using FileMaker in the AutoTyp project[43]. However, over time, the set of tables required to maintain such a database becomes quite complex—and performing basic queries like, "Give me all the primary data on language X" becomes difficult because the data may be scattered across a large number of tables, each of which must be known about for the search to give the desired results. Furthermore, the database structure is brittle. Reworking data in one table may break crucial links to other tables, and creating a new table may require that the data in previously existing tables be "refactored" into a new set of tables.

The issue here is a simple one: Typological work, when restricted one narrow research project, will often produce data easily modeled tabularly, but, when it steps outside such bounds and moves into long-term, collaborative projects where data will continually be added and classifications revised, the content to be encoded in a database quickly takes on more graph-like properties, where the data is better understood as a network connecting individual data points to each other rather than as a series interrelated of tables.

In sum, for simple data sets, it seems clear that tabular representations, of the sort found in relational database systems, may be quite desirable. The tool support is excellent, and the model is one all linguists should be comfortable with. But, for more complex data sets, graph-based representations will often be more suitable. And when such representations are desired, RDF databases offer one way to encode them.

A final advantage to RDF systems worth mentioning here involves the uses of reification, discussed in section 3.5.2. Reification, in effect, allows one to encode arbitrary metadata on *any* piece of information in the database. Since triples referring to reified triples can themselves be reified, it is possible to further encode meta-metadata, meta-meta-metadata, etc. That is, the possibilities to qualify the nature of any statement in the database is unlimited.

Compare this to tabularly-organized data. Consider, for example, table 1. One can easily assign metadata to the entire table, to a column of the table, and to a record in the table in a relational database system. But, how can one enter metadata specific to the claim of the table that Turkish has SOV word order? Suppose, for example, one wants to enter a source for this claim. One could

[43] http://www.uni-leipzig.de/˜autotyp/

add a source column—but then one needs to specify the source applies only to the word order data since, presumably, another source was used for the consonant inventory data. One could either split the data into multiple tables to encode this or use multiple source columns. Now, suppose one wants to further indicate that the language used to describe the consonant inventory for English and Irish is English, while for Turkish it is French. This, too, could be done relationally, of course, but, it should be clear that, as metadata needs become elaborated, it quickly gets more and more complicated to create and maintain all the needed relations holding among the data points using solely tabular representations.

LANGUAGE	CODE	WORD ORDER	CONSONANT INVENTORY
English	eng	SVO	average
Irish	gle	VSO	large
Turkish	tur	SOV	moyen

Table 1: Word order in three languages (sample table)

In an RDF representation of data like that in table 1, however, adding a wide variety of different kinds of metadata is trivial: Treat each cell in the table as a triple and add metadata, as needed, to each triple using reification. Of course, if one does not need a high-degree of metadata precision, then such a mechanism is not needed. But, we expect that, for certain applications, linguists will find the ability to assign metadata at a high level of precision quite useful, and RDF databases are well suited to this task.

None of the points raised here should be interpreted as an argument that linguists should abandon relational databases in favor of RDF databases—clearly, this would be undesirable. Our goal here was simply to discuss cases where we believe the graph-based model of RDF databases may be a more natural choice for encoding data than the table-based relational database model. We have chosen to discuss these issues in this context because RDF technology is fairly new, and we believed other members of computer-assisted linguistics community might find this sort of commentary valuable if they are considering adopting an RDF database for their projects, not to discourage people from making use of relational database technologies for tasks they are well designed for.

4 Conclusion

In this paper, we have outlined a database structure which we believe is well-suited for distinguishing between stable, non-contested content and unstable, contested content in ways which ensure that non-contested content is not unduly dependent on contested content, while still allowing non-contested and contested content to be brought together where desirable. The core of the conceptual model was to view non-contested content as encoded in stable nodes, on which very little organizational structure is imposed, and to view contested content as relationships holding among these nodes. This conceptual model has been implemented by using an object-oriented database for most nodal content and an RDF database for encoding relationships. However, once RDF tools were put into place, we found it made sense for some stable content to also be placed within an RDF database.

The data that was the focus of this paper—genealogical relationships among languages—is not of direct concern to most documentary and descriptive projects. However, we believe that the model and some aspects of the implementation proposed here may be applicable to a large number of problems involving contested content. Consider, for example, the annotation of primary text data for grammatical information. A sequence from a text could be conceptualized as a database node—since it is stable and non-contested. Furthermore, a large number of labels for grammatical categories exist, whose role in annotation is also not contested.[44] These could also be conceived of as database nodes. However, the association of a particular sequence from a text to a grammatical label is likely to be contested (perhaps even within the mind of a single annotator). Such associations, therefore, could be stored as relationships between text sequences and grammatical labels. Thus, the unstable analysis of a text could be separated from the stable text and a stable set of category labels.

Of course, much of what we are discussing in this scenario is central to the vision on which the GOLD Community—an offshoot of E-MELD—is based. Here, we are not offering a whole new set of ideals to aim for. Rather, we have described an implemented database model which, ultimately, we believe could play a role in attaining the ultimate aims of initiatives like E-MELD with respect to resource interoperability.

[44] Of course, this is not to say other aspects of those categories are not contested.