A Model and Architecture for Conceptualized Data Annotations

Michael Gertz Kai-Uwe Sattler
Department of Computer Science, University of California, Davis
One Shields Avenue, Davis, CA 95616-8562, USA
{gertz|sattler}@cs.ucdavis.edu

November 2001

Abstract

In many collaborative research environments, in particular computational sciences, novel tools and techniques allow researchers to generate data from experiments and observations at a staggering rate. Researchers in these areas are now facing the strong need for querying, sharing and exchanging the data in a uniform and transparent fashion to further leverage their findings. However, due to the nature of the various types of heterogeneous data and lack of local and global database schema structures, standard data integration approaches fail or are not applicable.

A viable solution to this problem is the extensive use of metadata. In this paper, we present the model and realization of a metadata management systems suitable for such research environments. The core component of the model are *conceptualized data annotations* that allow researchers to associate well-defined, agreed upon metadata, called *concepts*, with possibly remote Web accessible data sources. Annotations provide rich linkage structures between domain specific conceptual structures and fine-grained document components, including text and image data. We detail how annotations are managed and used in different data retrieval scenarios in the context of an application in the Neurosciences. We also present the realization of a general architecture for the proposed metadata information systems and outline some of the services embedded in this system.

Keywords: Metadata, Web Content Management, IS Infrastructure, Web Services

1 Introduction

In the past few years, many *computational sciences* such as Astrophysics, Biology, and Neurosciences, have witnessed a drastic increase in novel technologies and tools that help researchers to generate, record, and analyze various types of data from experiments. Researchers collaborating in such areas are now facing the problem of sharing, exchanging, and querying the highly

heterogeneous forms of data in a transparent and uniform fashion [Kar96, MPE98, WMG+99] to further leverage their findings. Because of the heterogeneity of the data and lack of local and global database like structures, however, standard approaches and architecture to data integration (e.g., [BE96, ERS99, Wie92]) are often not sufficient or inapplicable. Thus other information system (IS) methodologies and models need to be devised that help researchers in managing and accessing the distributed and heterogeneous data.

As several studies and reports indicate, e.g., [BFM+97, WMG+99], in collaborative research environments as the above, metadata play a crucial role. The types of metadata researchers are concerned with include not only information about what data is managed by different groups and how they can be accessed, but in particular forms of metadata that add *value* to the existing data. These forms of metadata typically arise when researchers interpret data, compare it with other data, or apply computational tools to the data. Such type of metadata, termed as *content descriptive metadata* [KS98, She99], typically occurs in the form of the description of particular concepts (or features) that are of interest for a researcher or experiment in a particular application domain. Most importantly, these concepts are "manually" associated with the data by researchers since they typically cannot be extracted automatically from the data, an aspect very common in cases where images build the majority of data.

In this paper, we refer to such type of metadata as *data annotations*. Annotations are made by researchers and can range from simple free-form texts to agreed upon, template like structures that researchers fill out to record their findings. These aspects motivate a strong need for an IS concept that allows researchers to add annotations to (remote) data sources and Web documents, share annotations with other researchers, and to utilize the annotations for different data retrieval scenarios. Most importantly and core concern of our approach, annotations need to be *controlled* and *external*. Although free-form annotations (e.g., a phrase or paragraph, similar to PostIt Notes) can be useful to a certain extend, they do not provide a basis for a meaningful usage of metadata for data retrieval. Ontological commitments based on some kind of standard vocabulary, taxonomy or rich ontology like structure play a crucial role in each of the above application domains in order to facilitate an effective usage of annotations. Only if there is some kind agreement on what features of data are of interest and what general properties such concepts (should) exhibit, annotations can build a core component of an IS to record, share, manage, and query controlled metadata in an effective manner. Annotations also should be external. That is, users should be able to annotate remote Web documents. Thus, a respective annotation model and system has to provide users with means to associate different conceptualized annotations with documents without having to modify the documents. It is our objective to develop an information system infrastructure that (1) realizes a rich and yet easy to implement model allowing users to annotate (possibly remote) data using domain specific *concepts* and relationships among concepts, and (2) provides researchers with a collection of basic services and tools to manage, share, and query concepts and annotations.

Related Work. In the past few years, there has been a great amount of work on models and methodologies to semantically enrich the Web (see www.semanticweb.org for an extensive overview). The major focus in these works is on building semantic rich and expressive ontology models that allow users to specify domain knowledge. The most prominent approaches in this

area the Ontobroker project [DEFS99, FAD+99], SHOE [HH00], and the Topic Maps standard [ISO99, Pep00]. We consider these ontology-centric works as orthogonal to our annotation-centric approach and they thus can be realized as additional services in our IS. Also, most of these work do not put much emphasis on how remote Web documents can be annotated by different users at a fine-level of granularity. We consider the need for external and fine-grained annotations as essential and appropriately include these aspects in our model. Furthermore, whereas the above approaches concentrate on querying ontologies using, e.g., RDF-based languages, our focus is to have a simple, expressive, and easy to implement language that allows to query all three components, concepts, annotations, and Web data in a uniform fashion.

At the other end of the spectrum, several systems have been proposed that provide users with means to annotate data. This includes the multivalent document approach [PW97] as well as some commercial systems (see, e.g., [Gar99, HLO99] for an overview). However, none of the proposed approaches support either conceptualized annotations or a query and service framework for annotations.

The initial idea of external, conceptualized annotations has been presented in [BG01]. In this paper, however, neither a coherent conceptual model for annotations, concepts, and Web documents nor a query framework has been given. In this work, besides presenting the model and query framework, we also present an IS architecture for realizing different data retrieval services.

Structure of the Paper. In the following section, we present more detailed examples of application scenarios in the context of the Neurosciences. These examples further motivate the need for conceptualized data annotations and illustrate the services a respective IS infrastructure should support. The annotation model allowing users to specify links between Web-accessible data and concepts pertinent to an application domain is detailed in Section 3. In Section 4, we detail several consistency aspects in the context of creating and modifying concepts and data annotations. After a discussion of mapping the annotation model to a relational database in Section 5, in Section 6 we present the realization of our prototype and services. Section 7 concludes this paper with a summary and outline of future work.

2 Application Scenarios and Architectural Components

Our study of conceptualized annotations takes place in the context of the Human Brain Project Initiative [NIH, KH97, UCD], a sub-discipline in the Neurosciences. The main interest in this project is to study neuroanatomical and neurobiological aspects of specific components of the human brain. Besides text data in form of journal articles and scientific reports, in this project, high resolution images obtained from photographs of brain slices and MRIs build the majority of the data generated and utilized by different research groups. In Section 2.1, we outline some usage scenarios in which conceptualized annotations are used to enrich Web-accessible data. Section 2.2 then details the implications the scenarios have on a respective IS infrastructure supporting such services.

2.1 Data Annotation Scenarios

Assume a research group publishes their data on the Web and that there is an IS component researchers use to register data they generate. Such a registry is simple to set up and can be based on a Dublin Core or RDF like metadata schema [DC0, LS99] for specifying location (e.g., URL), author information, and general content information about the data. Now assume a researcher browsing a newly generated set of images published by a research group (left browser window in Figure 1). In one of these images, the researcher identifies a region containing a cell structure of a known type A. Ideally, she would like to annotate this region and "link" it to a conceptual structure that has been defined in the specific research context. This structure defines a concept (similar to a metadata schema) that details general, agreed upon properties of the cell type A such as a definition, terms typically used to refer to the cell type, and other general properties of that concept. Through this linkage, the identified region in the image then can be understood as an *instance* of the concept and which then is filled out by the researcher (e.g., specific values for spatial properties of that region in this image). Assume the researcher knows about another group that deals with images from the same region, generated in the context of a different experiment by another group. She pulls up one such Web document which contains some images as well as some descriptive text (right browser window). The image in this document exhibits a region containing the cell type A. Again, she would like to link this region to the same concept used for the first image, thus specifying another instance of this concept with perhaps different values for the properties. The aspect of linking (annotating) these regions to a concepts and instantiating concept properties is indicated through the lines and boxes at the marked regions in Figure 1.

Many more complex annotation scenarios are concievable in the context of the above example. For instance, it should be possible to annotate text regions if the text discusses the cell type. Annotating text is useful in case journal articles and reports discuss a concept of interest. In this case, text data represent an instance of the concept describing that cell type. Also, if relationships between concepts are allowed (similar to relationship types in the ER model), such relationships can translate to relationships between annotated data. This is indicated in the above picture, where another cell type B typically co-occurs with cells of type A. This aspect is represented at the concept level through a relationship type concept "co-occurs-with" and at the document level through a link between two annotated regions. Such links among annotations can be understood as typed and thus semantic rich links, compared to simple HTML links.

While tools that help researcher in annotating data of heterogeneous forms at different levels of granularity are a core service provided by our IS infrastructure, other services are necessary to leverage the usage of annotations in a collaborative research environment and are discussed next.

2.2 Services

Several services are needed not only to manage concepts and annotations in a collaborative fashion but also to allow researchers to perform different data retrieval tasks based on annotations, annotated documents, and specified concepts. In the following, we outline some of these basic

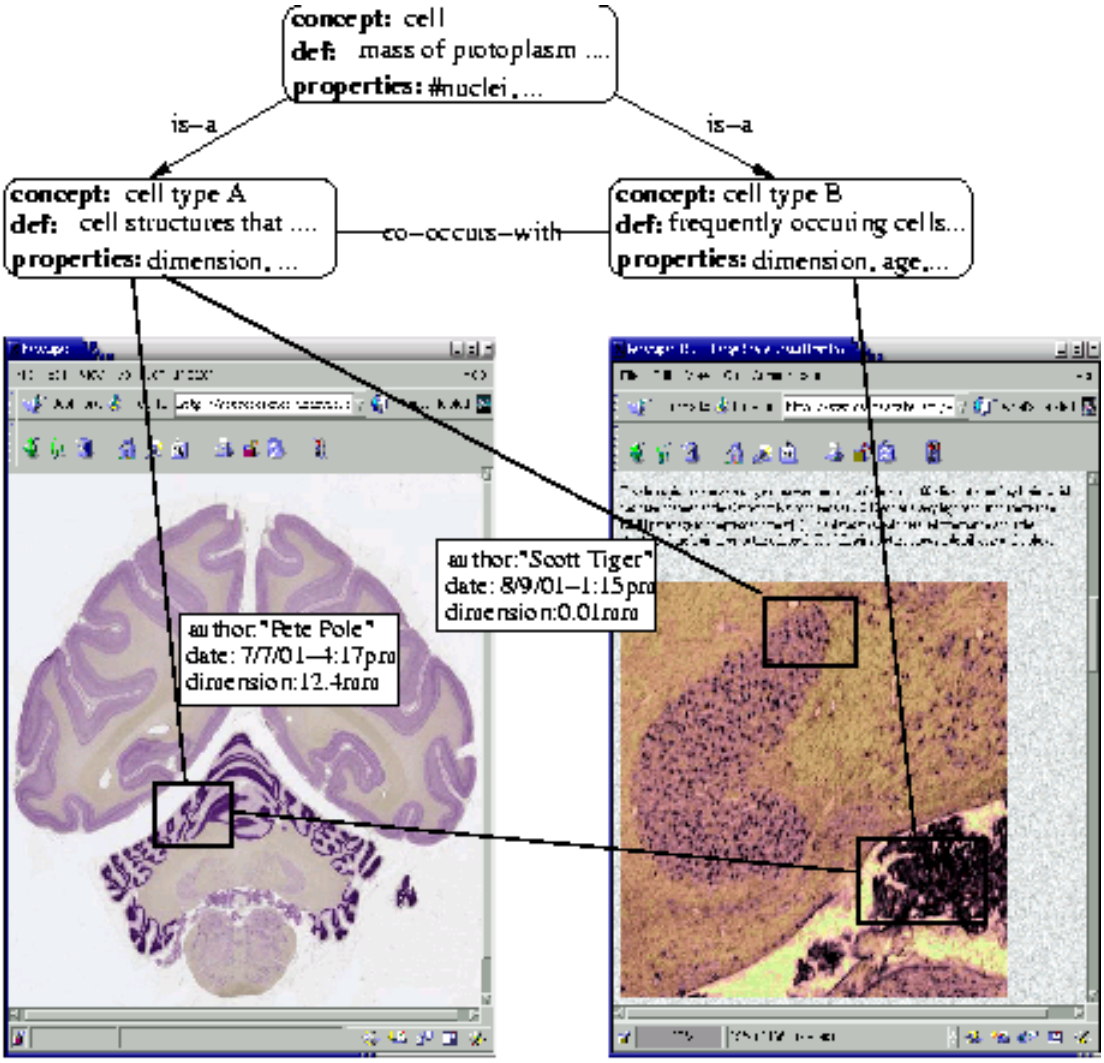

Figure 1: Scenario of Conceptualized Annotations for Web Data

services. The services are organized in a hierarchical fashion (see Figure 2). They are modular and thus easy to extend. We distinguish between concept-centric and annotation-centric services. Concept centric-services address aspects of modeling the domain knowledge, including the specification and management of concepts and have been extensively dealt with in the context of ontologies and semantic Web (see related work). Since the focus of this paper is on annotation-centric services, in our current prototype we only support a simple type of concept model, which is detailed in Section 3.

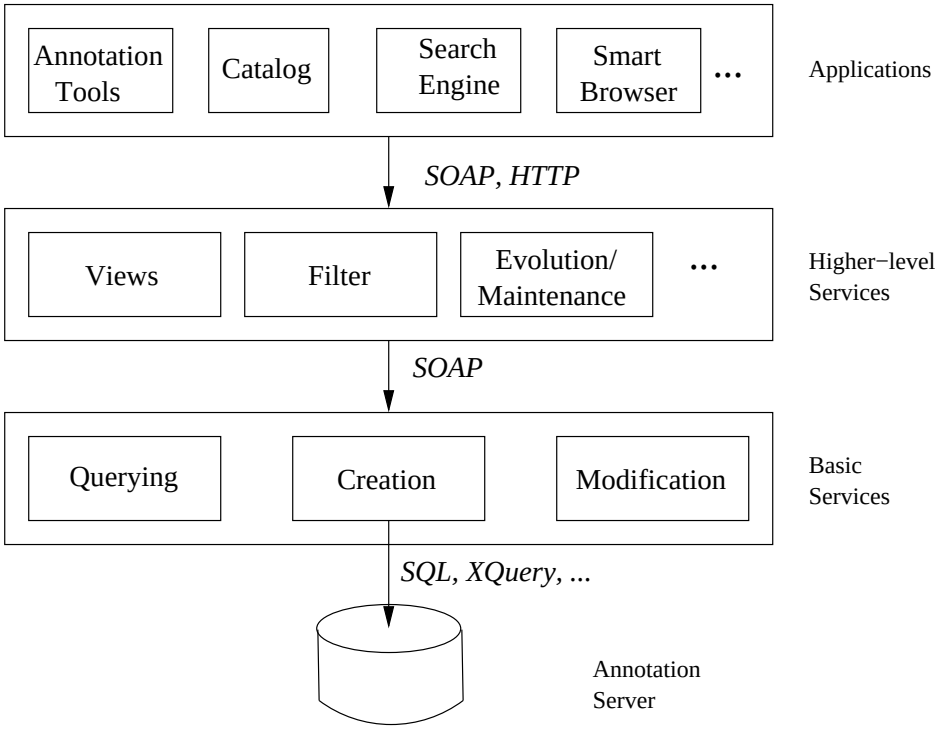

Figure 2: Conceptual Architecture of Annotation-centric IS Infrastructure

Annotation Tools: These tools allow users to upload Web documents and to associate concept instances (annotations) with the text and/or image data in these documents. Instance properties as well as document context data (e.g., image region information) are recorded in an annotation server. The Annotea service [KKHS01] together with the Amaya browser [W3C0] are an example of such a tool that allows user to annotate text documents. In our prototype, we employ this service for text data and a separate service for image data.

Annotation Server and base services Annotations representing "enriched link structures" between documents and concepts, are recorded in a logically centralized *annotation server*. Other services utilize this server to store and query annotations (and associated properties). In particular, a Web service, e.g., in form of a Web browser, that displays annotations made for a Web document utilizes this service to retrieve annotations associated with a Web document. Such retrieval operations can utilize user-specified filters, allowing researchers to specify to display

only annotations that have specific properties with respect to, e.g., underlying concepts, authors etc. This allows users to have different views on the same documents.

Smart Browsing: Given a collection of annotations that link Web documents to concepts, these linkage structures allow researchers to "discover" various types of semantic relationships between annotated documents. For example, documents that are annotated with the same concept (perhaps by different users) are probably semantically related. Also, if relationships between concepts are allowed, these relationships can be exploited at the document level to determine annotated documents that are related. That is, if the user uploads a document that has been annotated, the user should be able to traverse from the annotated document through the concepts back to other annotated documents. For example, if relationships between concepts provide a means to specify is-a relationships, documents that have been annotated with respective concepts can be used in data retrieval scenarios where a user is looking for similar documents. This functionality represents a more advanced, annotation-based variant of the "similar pages" model realized in Mozilla or the Internet Explorer.

Catalog: Concepts, relationships among concepts, as well as their linkage to Web documents via annotations naturally imply a categorization of annotated documents. Thus, by utilizing such concept-based relationships, a Yahoo-like catalog can be constructed for a heterogeneous, distributed collection of Web documents. The computation of such categories is trivial as long as *is-a* relationships are supported at the concept level. Other types of relationships between concepts, e.g., spatial, temporal, or other forms of general semantic relationships [TRW+97], can be used to provide the users with different ways to categorize annotated documents at a domain specific, conceptual level.

Query Engine: Since both concepts and annotations are assumed to be managed by the annotation server, they can easily be queried. Several query scenarios need and can be supported in this context. For example, it should be possible to obtain all documents (or parts thereof) that have been annotated using a specific concept. Based on the general idea that annotations are used for describing and detailing the content of documents using well-defined, agreed upon concepts, this aspect results in a *query engine* for metadata. Further types of queries include finding (combinations of) annotations and concepts that satisfy user-specified conditions. In addition, due to possible relationships between concepts and annotations, the user can change the query context between these two realms and find, for example, other annotations (and in this way annotated documents) that have been annotated with the same or a related concept.

In summary, the query engine scenario described above is a generalized variant of the other ones because all of the other services can be built on top of it. Thus, an annotation and query server providing functionality to (1) manage concepts, annotations, relationships among each other and to documents, and (2) to query both concepts and annotations is able to support all of the applications described above. In order to build domain-specific higher-level services and applications all the services should provide open interfaces using standard-conforming technologies and be Web accessible. One way to achieve this is by implementing them as Web Services. We will discuss this aspect in Section 6.

3 Annotation Graph Model

In order to realize the services described in the previous section, a solid formal model for representing and querying concepts, annotations, and Web documents is essential. In the following, we present an *annotation graph model* that addresses the above needs in terms of expressiveness, extensibility, and ease in implementation. In this model, concepts, annotations, and documents are represented as different types of nodes and edges between nodes describe respective relationships such as how concepts are related and documents are annotated using concepts.

In Section 3.1, we introduce the basic core components of the annotation graph model. Section 3.2 details the semantics of different types of edges in this model. In Section 3.3, we discuss various types of query operations that can be performed on instances of the annotation graph model. Section 4 finally outlines data consistency aspect regarding the conceptualized annotations.

3.1 Base Components

Assume a set $T = \{\mathit{String}, \mathit{Int}, \mathit{Date}, \dots\}$ describing a set of simple data types. Let $\mathrm{dom}(T)$ denotes the domain, i.e., the set of all possible values for T. In the annotation graph model, a property of a node is defined by an identifier and a type $\mathit{PDef} = \mathit{String} \times T$. An instantiation of a property consists of an identifier and a value $\mathit{PVal} = \mathit{String} \times \mathrm{dom}(T)$.

As outlined in the previous sections, concepts provide templates for annotations that are associated with document. In our model, concepts are represented by a simple form of *concept nodes*. More complex concept specifications are conceivable but will not be discussed in this paper (see related work in the introduction). We assume that each concept node has an identifier, several descriptive terms used to name the concept and a set of property definitions. We denote the set of all concepts by $\mathcal{C}$. Let be $\mathit{Def} \subseteq \mathit{String}$ be a natural language definitions that associate an agreed upon, well-defined meaning with the concept and let $\mathit{Term} \subseteq \mathit{String}$ be a set of terms in form of a word or phrase that are typically used in a specific application context to name (and query) a concept. Then

$$\mathcal{C} \subseteq \mathit{Term} \times \mathit{Def} \times \mathbb{P}\,\mathit{Term} \times \mathbb{P}\,\mathit{PVal}$$

For a concept $(\mathit{cname}, \mathit{def}, \mathit{terms}, \mathit{pdefs}) \in \mathcal{C}$,

- *cname* is the name of the concept (typically a preferred term),
- *def* is a textual description or definition of the concept in natural language,
- *terms* is a set of synonyms used to name the concept, and
- *pdefs* is a set of property definitions.

In this respect, a concept definition is very similar to a class definition in the context of object-oriented modeling.

The second type of node in our graph model represents Web accessible documents. Documents are identified by a URI (Uniform Resource Identifier) and have a title. Other properties

of documents as they are recorded in the previously mentioned data registry component can be included in the description of a document as well. Both URI and title are assumed to be of type *String*. We denote the set of all Web documents by $\mathcal{D}$.

Finally, annotation nodes provide the basis to specify links between concepts and documents. The set $\mathcal{A}$ of annotations is defined as

$$\mathcal{A} \subseteq \mathit{String} \times \mathit{Date} \times \mathbb{P}\,\mathit{PVal}$$

For a tuple $(\mathit{creator}, \mathit{created}, \mathit{pvals}) \in \mathcal{A}$,

- *creator* is the author of the annotation,
- *created* is the creation date/time,
- *pvals* is a set property instantiations.

The set of all nodes $\mathcal{V}$ in an annotation graph is now defined as $\mathcal{V} = \mathcal{A} \cup \mathcal{C} \cup \mathcal{D}$. Links between nodes are represented as directed, typed edges, to which property instantiations are optionally assigned. The types of edges are drawn from the specified concepts (see below) and the property instantiations are determined by the associated concept. Finally, the set $\mathcal{E}$ of all edges is define as

$$\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{C} \times \mathbb{P}\,\mathit{PVal}$$

The meaning of the components of an edge $(\mathit{from}, \mathit{to}, \mathit{type}, \mathit{pvals}) \in \mathcal{E}$ is as follows:

- the edge connects the node *from* with the node *to* (in this direction),
- with the edge the concept *type* is associated,
- *pvals* is a set property instantiations.

For example, to represent an "is-a" relationship between two concepts c_{super} and c_{sub} we define an edge $(c_{sub}, c_{super}, \mathit{isA}, \perp)$ where *isA* is the identifier of a defined concept. In an implementation of this model, the kind of nodes connected by an edge should be further restricted, but this is not addressed in this basic model.

Using these definitions, our annotation graph model comprises both the metadata schema level components (concepts) and instance level components (annotations and documents). An instance of the model defined by one or more users is then represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

3.2 Relationship Types among Nodes

Naturally, nodes can be connected via edges of arbitrary types. However, only edges of certain types make sense in most cases. In order to deal with the special meaning of the different kinds of nodes and how they can be connected via edges, we introduce the following *default concepts* that are contained in any specification of a collection of concepts. An instance of annotation graph illustrating the different relationship types is illustrated in Figure 3.

- $\langle\rangle$ denotes an unnamed concept, specifying a simple edge type "references",

- *annotates* is used to represent edges from annotations to documents. If an annotation is created for a document, a link referring to this concept is defined between both of them. Since we are interested in fine-grained annotations, e.g., regions in an image or text fragments such a paragraphs, we assume a set of of *document context descriptions*. Such a description eventually comprises information about the document context in which an annotation occurs and is modeled as a property of the relationship. In our current system, we employ two types of descriptions. For text-based documents, we employ a tree-structured document model. In particular, we employ a transformation mechanism to handle HTML and plain text documents as XML document. Thus, XPath expressions [CD99] are used as document context descriptors. For image based data and annotations, we use region, point, and scale information to encode descriptions for respective fine-grained annotations. A similar approach can be found in [Ino].

- *annotatedBy* is a concept for representing the inverse of *annotates*. So, it holds[1]:

  $\forall e \in \mathcal{E} : e.\textit{rel} = \textit{annotates} \rightarrow e.\textit{from} \in \mathcal{A} \wedge e.\textit{to} \in \mathcal{D} \wedge$
  $\exists(e.\textit{to}, e.\textit{from}, \textit{annotatedBy}) \in \mathcal{E}$ and
  $\forall e \in \mathcal{E} : e.\textit{rel} = \textit{annotatedBy} \rightarrow e.\textit{from} \in \mathcal{D} \wedge e.\textit{to} \in \mathcal{A} \wedge$
  $\exists(e.\textit{to}, e.\textit{from}, \textit{annotates}) \in \mathcal{E}$

- the concept *ofConcept* represents the fact that an annotation is based on a certain concept, i.e., assigns the concept to the annotated document and instantiates the properties defined by this concept.

- *hasAnnotation* describes the inverse relationship to *ofConcept*. Again, it holds:

  $\forall e \in \mathcal{E} : e.\textit{rel} = \textit{ofConcept} \rightarrow e.\textit{from} \in \mathcal{A} \wedge e.\textit{to} \in \mathcal{C} \wedge$
  $\exists(e.\textit{to}, e.\textit{from}, \textit{hasAnnotation}) \in \mathcal{E}$
  $\forall(n_A, v_A) \in e.\textit{from.pvals} : \exists(n_C, t_C) \in e.\textit{to.pdefs} \wedge n_A = n_C \wedge$
  $v_A \in \mathrm{dom}(t_C)$
  $\forall e \in \mathcal{E} : e.\textit{rel} = \textit{hasAnnotation} \rightarrow e.\textit{from} \in \mathcal{C} \wedge e.\textit{to} \in \mathcal{A} \wedge$
  $\exists(e.\textit{to}, e.\textit{from}, \textit{ofConcept}) \in \mathcal{E}$

  In addition, each annotation $a \in \mathcal{A}$ must be related to a concept:

  $$\forall a \in \mathcal{A} : \exists c \in \mathcal{C} \wedge (a, c, \textit{ofConcept}) \in \mathcal{E}$$

- *isA* denotes an "is-a" relationship between concepts implementing inheritance of property definitions:

  $\forall e \in \mathcal{E} : e.\textit{rel} = \textit{isA} \rightarrow e.\textit{from}, e.\textit{to} \in \mathcal{C} \wedge e.\textit{from.pdefs} \supseteq e.\textit{to.pdefs}$

In Fig. 3 different relationship concepts are shown for the scenario from Fig. 1. For example, the edge between concept C_3 ("cell type A") and C_2 ("cell") represents an "is-a" relationship

[1] We use the notation $t.e_i$ to denote the element e_i of a tuple $t = (e_1, e_2, \ldots, e_n)$ for $1 \leq i \leq n$.

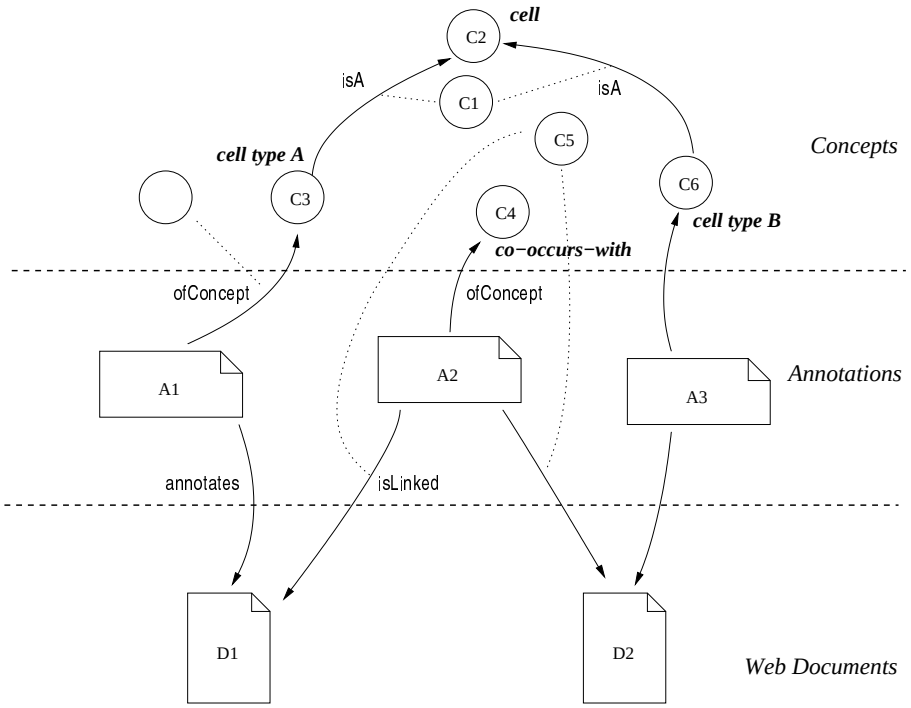

Figure 3: Instance of an Annotation Graph

and the edge between annotation A_1 and document D_1 denotes that A_1 is an annotation to this document. Finally, annotation A_2 links two different documents. Here, the meaning of this linkage is expressed by the concept C_4 ("co-occurs with") and the individual participation of the documents is modeled by the concept C_5. Note that inverse relationships like *annotatedBy* etc. are omitted for better legibility.

3.3 Query Operations

Querying and navigating an annotation graph is supported by two kinds of operations: selection and path traversal. The input for a selection operation is always a homogeneous set: either one of the basic sets $\mathcal{A}$, $\mathcal{C}$ or $\mathcal{D}$ (but not a union of them) or a derived set resulting from a prior operation. Let be S one of the sets $\mathcal{A}$, $\mathcal{C}$ or $\mathcal{D}$ and $P(s)$ a predicate on $s \in S$. Then the selection operation σ_P is defined as follows:

$$\sigma_P(S) = \{s \mid s \in S \wedge P(s)\}$$

Here, a predicate is a boolean expression made up of a number of clauses like *prop* <*op*> *value* which can be connected by the boolean operators. In addition, path expressions in the form $rel_1.rel_2 \ldots rel_n.prop$ indicating the traversal of edges of concepts rel_1, rel_2 etc. from the current node to the property *prop* of the target node are allowed as long the result is a single valued expression. Accessing non-existing properties evaluates always to false.

Path traversal enables following links between nodes of the graph. Given a start node v_s and a relationship rel by means of a concept, the operation ϕ_{rel} returns the set of target nodes based on existing edges:

$$\phi_{\mathit{rel}}(v_s) = \{v_t \mid (v_s, v_t, \mathit{rel}) \in \mathcal{E}\}$$

Because we mainly have to deal with sets of nodes in query expressions this operation is also defined on a set of nodes $V \in \mathcal{V}$:

$$\Phi_{\mathit{rel}}(V) = \{v_t \mid \forall v_s \in V : (v_s, v_t, \mathit{rel}) \in \mathcal{E}\}$$

A special kind if the path traversal operation is the operation for computing the transitive closure. This extends ϕ_{rel} by traversing the path indicated by the relationship as long as edges can be found which were not already visited. The final set of all involved nodes is returned as the result:

$$\begin{aligned} \phi^+_{\mathit{rel}}(v_s) \quad = \quad & \{v_t \mid (v_s, v_t, \mathit{rel}) \in \mathcal{E} \vee \\ & \exists v_i \in \phi^+_{\mathit{rel}}(v_s) : (v_i, v_t, \mathit{rel}) \in \mathcal{E}\} \end{aligned}$$

As for ϕ_{rel} this operation is defined on a set of nodes, too:

$$\begin{aligned} \Phi^+_{\mathit{rel}}(V) \quad = \quad & \{v_t \mid \forall v_s \in V : (v_s, v_t, \mathit{rel}) \in \mathcal{E} \vee \\ & \exists v_i \in \Phi^+_{\mathit{rel}}(v_s) : (v_i, v_t, \mathit{rel}) \in \mathcal{E}\} \end{aligned}$$

Based on these operations, query expressions for selecting nodes and traversing edges can be formulated. Here, the initial set of nodes for traversals always has to be obtained by applying a selection on one of the basic sets $\mathcal{A}$, $\mathcal{C}$ or $\mathcal{D}$. From this, following the special relationships *ofConcept*, *annotates* etc. allows to go to another type of nodes (Fig. 3). In this way, starting from a document specified by a URI, the associated annotations can be found. If now the *ofConcept* relationships are followed, we are able to obtain the concepts and by going "downwards" from there to the annotations and documents, we can find "similar" document (fragments), i.e., documents that are annotated by the same concepts:

$$\Phi_{\mathit{annotates}}(\Phi_{\mathit{hasAnnotation}}(\Phi_{\mathit{ofConcept}}(\Phi_{\mathit{annotatedBy}}(\sigma_{uri=\ldots}(\mathcal{D})))))$$

This query expression could be further extended by considering the *isA* relationships of the involved concepts:

$$\Phi_{\mathit{annotates}}(\Phi_{\mathit{hasAnnotation}}(\Phi^+_{isA}(\Phi_{\mathit{ofConcept}}(\Phi_{\mathit{annotatedBy}}(\sigma_{uri=\ldots}(\mathcal{D})))))$$

The queries shown above always return a set of nodes representing documents. In contrast, in the following sets of annotations are returned:

$$\Phi_{\mathit{annotatedBy}}(\sigma_{\mathit{uri}=\ldots}(\mathcal{D}))$$
$$\Phi_{\mathit{hasAnnotations}}(\sigma_{\mathit{cname}=\ldots}(\mathcal{C}))$$

The first query retrieves only the annotations associated with the document that is identified by a certain URI. The query in the second example returns annotations of a given concept. As

a further example, the following query returns the document(s) that is/are linked to a given document by an annotation of a certain concept C:

$$\Phi_{annotates}(\sigma_{ofConcept.cname=C}(\Phi_{annotatedBy}(\sigma_{uri=...}(\mathcal{D}))))$$

Here, starting with the document for a given URI, the annotations are obtained. The set of annotations is restricted to the annotations of concept C and for these, the set of annotated documents is selected. Of course, queries can utilize not only the predefined relationship concepts but also user-defined relationships.

4 Consistency of Annotations

Allowing users and research groups to introduce concepts and annotate documents naturally can lead to certain types of redundancies and inconsistencies in an annotation graph. In order for conceptualized data annotations to be useful for different data retrieval tasks, appropriate methods and mechanisms need to be devised that ensure a certain degree of non-redundancy and consistency among data annotations.

In this section, we discuss how such mechanisms have been or will be realized in our prototype of a data annotation system. In Section 4.1, we first detail the problem setting and goals in constructing an annotation graph in a collaborative fashion. In Section 4.2, we then discuss some consistency checking methods that exclusively operate on the concept level. In Section 4.3, we finally detail how properties of annotations can be employed for different redundancy and consistency checking mechanisms.

4.1 Problem Setting

An annotation graph typically is constructed in a collaborative fashion. Over time, users and research groups introduce and modify concepts and annotations. Thus, an annotation graph utilized in different data retrieval tasks is a very dynamic structure. In order for the annotations to be useful in these tasks, services need to be provided that check for a certain degree of non-redundancy and consistency in terms of how concepts and annotations have been used by different users.

Consider the scenario where some agreed upon base concepts have initially been introduced by some kind of expert committee, and these concepts are used by different users in annotating documents. Different users might have a different focus and thus it is reasonable that, e.g., different concepts are used to annotate the same data. Also, often there exists no initial pool of concepts users can choose from, and thus every user introduces her own concepts and naming conventions. This is a very common scenario in the application domain we are focusing on. Because of new technologies, neuroanatomical features in brain structures can be detected and analyzed at a level of granularity not possible before. Different people are investigating the newly generated data and thus often introduce semantically equivalent concepts that describe the detected structures. The problem becomes even more complicated if changes on concepts, concept relationships, and data annotation occur frequently.

The problem of redundancies and inconsistencies among concepts is a well known problem and has been studied extensively for ontology like structures in, e.g., [Gru93, SPKR96, FFR96]. Also, in the medical domain where medical and clinical terminology systems are very popular and extensively used, the problem of inconsistent and redundant terminologies often is of particular concern, especially in the context of change management [COS98, OSS99, Oli98].

For a complex collaborative research environment, it is natural that different users have different views on the data. That is, the research focus and thus interpreted features in the data can vary. Thus, we assume a *view model* on top of an annotation graph. For each user, her view consists of all concepts and annotations that she has specified but can also include views of other users that are made available and are described in a respective service. Different types of views can easily be specified using the query model introduced in the previous section. As a minimum, one should require that at least concepts and annotations within one view are non-redundant and consistent. In case possible inconsistencies occur through the specification of new concepts or annotations or existing ones are modified, *change policies* can be specified by a user and help in (semi-automatically) determining and resolving possible inconsistencies. In the sections below, we will outline some change policies in the context of different types of modifications to concepts and annotations.

It is our claim that while in traditional ontology or terminology based systems it is hard to deal with redundant and inconsistent concepts, through the usage of data annotations possible inconsistencies can be better identified and dealt with. The main reason for this is that since concepts are actually used to describe data through annotations, users can inspect the annotations and negotiate the correct usage of the underlying concepts. That is, one actually has a many more data (in form of annotations and documents) to better compare concepts and concept relationships.

4.2 Concept Level Mechanisms

As indicated above, devising methods and implementing mechanisms that guard against redundant and/or inconsistent concepts is a hard problem. One major reason for this is that it is hard to exactly define what a redundant or inconsistent concept actually is. In our work we thus take a liberal approach where possible inconsistencies are simply pointed out to the user upon concept creation or modification. The user then can choose among possible strategies, so-called *modification policies*, to resolve the conflict. In the following, we assume that a user is operating under a specific view and that checks occur in the context of this view and other views specified by the user. In the sequel, we consider the following types of modifications:

- creation of a concept,
- specification of a relationship type among concepts,
- modification of a concept (including its relationships), and
- deletion of a concept.

Intuitively the introduction of a new concept without any relationship to existing other concepts does not cause any direct consistency problem. However, the new concept can be *semantically similar* to an existing concept and thus may be redundant. There are several possibilities to define and check for semantic similarity among two concepts by investigating the similarity among terms and attributes of the two concepts. Assume a newly defined concept c and an existing concept c'. If for the concept c a term has been specified such that this term is also used for another concept, then this can be indicated to the user. Different strengths of similarity can be defined depending on how many terms are related. The more terms of c are related to c' the similar the two concepts are. Stemming mechanisms for terms can be employed in such checking scenarios. Analogously, if one or more attributes have been specified for c such that these attributes are also used for c', then this can indicate a possibly redundant concept. The more attributes are shared with another concept, the similar the two concepts are in this case. In interesting aspect in this case is that if the attributes of c form a subset of the attributes associated with c', then c can be indicated to the user as a superconcept of c', related through an "is-a" relationship. Again, the strength of similarity in terms of how many attributes need to be related in order for the system to alert the user can be specified by the user.

If the user introduces a relationship between two existing concepts, say c_1 and c_2, other existing relationships between the two concepts should be indicated to the user. In case the relationship introduced is a "is-a" relationship, the system can easily check whether there is no cycle or contradiction between c_1 and c_2 such that only "is-a" relationships are involved. Similar mechanisms can be devised for other relationship type concepts that support inheritance. In general, for specifying relationships between two concepts it is helpful to show the user the context of the two concepts in terms of what other concepts are related to these concepts. In this respect, a tool needs to be devised that graphically represents the context (subgraph in an annotation graph) of where the specification of the new relationship takes place.

A modification of a concept typically involves the change of terms or attributes associated with the concept. In both cases, an approach similar to the creation of a new concept can be taken in order to check for existing semantically similar concepts. Thus, the change policies can be devised in a fashion similar to those for concept creation.

The deletion of a concept that is not related to any other concept only causes all annotations based on this concept to be deleted as well. If the concept to be deleted is related to other concepts, however, different scenarios are conceivable. We detail one such scenario for the case where a concept c has an "is-a" relationship with other concepts, that is, c is embedded in a "is-a" hierarchy. Assume c has a a superconcept c'. In case c is deleted, all annotations can be "re-linked" to the concept c'. This type of change policy requires that all instantiations of attributes c has in addition to those from c' are deleted as well. In case c does not have a superconcept but a subconcept, then another change policy would be to ask the user whether the annotation(s) to be deleted can be refined to instances of the concept c'. In both cases, the metadata associated with the documents are preserved.

4.3 Annotation Level Mechanisms

As indicated in Section 4.1, we claim that by using conceptualized annotations for describing data, we can perform more extensive checks for the consistent usage of concepts. In the following, we are mainly concerned with two cases of modifying annotations:

- the creation of an annotation, and
- the deletion of an annotation.

Assume a user specifies an annotation a_1, based on a concept c_1 for a document d_1 such that the context of the annotation in d_1 is df_1. Recall that for a text document, df_1 is a specification of a document fragment in terms of an XPath expression, which addresses, for example, a specific section or paragraph. For an image, df_1 describes a region in the image in form of a polygon. Depending on the view chosen by the user, other annotations (made by other users) may exist for the document d_1. Consistency and redundancy checks are now based on how the new annotation a_1 is related to the existing annotation in terms of context and underlying concepts. Note that all information about existing annotation, including context information and underlying concepts, can easily be obtained through respective queries against an instance of the annotation graph (or rather against a view thereof).

Let df_2 be the context of an existing annotation a_2 for the document d_1. We now check whether df_1 and df_2 are comparable, and if so, of what type their relationship is. That is, we would like to determine whether df_1 is a refinement of df_2, or vice versa. For a text document, this means that df_1 is a refinement of df_2 if df_1 addresses a document fragment in the document fragment specified by df_2. Since the XPath expressions used for document fragments are simple path expressions (i.e., they do not contain any wildcards and address exactly one node in the document), such a property can easily be checked. For images, respective procedures check for the overlap or containment of polygons describing regions in the images. There are three cases to consider:

1. df_1 and df_2 are not comparable. In this case, there is no direct relationship between the two annotations a_1 and a_2. However, the user can choose to display all annotations that have been made for the document d_1 (under this view) such that these annotations are "close" to the newly specified annotation. Querying respective underlying concepts of these annotations can help the user in better understanding the semantic context in which she made the annotation.

2. $df_1 \leq df_2$. If the two context are equal, ideally the two annotations a_1 and a_2 should be based on the same concept. If the two concepts are equal, then the instantiation of the properties should be the same as well in which case the newly defined concept would be redundant. If not, this is displayed to the user. If the underlying concepts are different, this is displayed to the user as well, and the user can utilized this information to evaluate her annotation or to obtain information about how the data is interpreted by another user.

   If df_1 is a refinement of df_2, then this aspect should be reflected by the underlying concepts as well. Ideally, since df_1 is more specific than df_2, the concept c_1 should be a subconcept

used for annotation a_2. If it turns out that c_1 is a superconcept of the concept used for a_2, then this is a possible case of an inconsistency and needs to be resolved appropriately by revising either annotation. In case there is no information about how the two concepts for a_1 and a_2 are related, evaluating the annotations a_1 and a_2 can lead to an introduction of such a respective relationship by the user.

3. $df_1 > df_2$, that is, the context of the new annotation is coarser than the context of the existing annotation. This scenario can be handled in a fashion similar to the previous case.

The above cases show that by comparing the newly specified annotation with existing annotations for the same document, the user is able to obtain various types of information. This includes in particular information about how other users interpret the data. It should be noted that in case an inconsistency is detected, e.g., an incorrect (or too coarse) concept has been used for an existing annotation, the user adding a new annotation cannot simply change existing annotations of other users. That is, a management infrastructure is necessary to negotiate changes among users (authors) of annotations and concept specifications. Such an infrastructure is not discussed in this paper and is left for future work.

We finally discuss how deletions of annotations can be dealt with. Although there is no need for special actions in case a user deletes an annotation, mechanisms and change policies are conceivable that try to preserve some information associated with the annotation to be deleted. For this, we exploit "is-a" relationships that might exist for the concept underlying the annotation to be deleted. Assume there is a concept c' which is related to the concept c_1 underlying an annotation a_1 to be deleted through an "is-a" relationship. Instead of a_1 being deleted, it can be "re-linked" to the concept c'. Such a change policy required the deletion of all attribute values of the annotation that are not covered by c' and might also require the extension of the context of a_1 to either a coarser document fragment (in case of a text document) or larger region (in case of an image).

5 Mapping the Graph to a Relational Database

While presenting the annotation framework in the previous section we made no assumptions on how concepts, annotations, and relationships are managed or queries are processed. In order to realize the services outlined in Section 2.2, appropriate storage and query components are required. Due to technical requirements of a specific project we have decided to use a relational DBMS as backend database system. Hence, the mapping of the graph structure to relations as well as transforming query expressions to SQL are necessary tasks.

The different kind of nodes and the edges of the graph are stored in relations with the following schemas[2]:

CONCEPT*(id, def)*
CONCEPTTERM*(id* → CONCEPT, *term)*

[2]Only the relevant schemas are shown.

CONCEPTPDEF*(id* → CONCEPT*, kind, name)*
CONCEPTRELSHIP*(id*→ CONCEPT*, fromTable, toTable)*
DOCUMENT *(id, uri, title)*
ANNOTATION*(id, creator, created)*
ANNOTATIONPVAL *(id*→ ANNOTATION*, name, val)*
RELSHIP *(id, from, to, rel*→ CONCEPT*)*
RELSHIPPVAL *(id*→ RELSHIP*, name, val)*

Concepts are stored in the normalized relations CONCEPT, CONCEPTTERM (for terms) and CONCEPTPDEF (for property definitions). If a concept describes a relationship, an additional tuple is created in the CONCEPTRELSHIP specifying which kind of nodes (in form of the relation names) are linked. Information about document is managed in the DOCUMENT relation and annotations are stored in the relation ANNOTATION together with the property values in ANNOTATIONPVAL. Finally, the relation RELSHIP is used for recording relationships together with relation RELSHIPPVAL for the property instantiations.

Based on this database schema the query expressions introduced in Section 3.3 have to be translated into SQL queries. This is done in two steps: first the expression is transformed into a relational algebra expression and then into a SQL query. In the following, we will consider only the first step, because deriving SQL queries from relational algebra expressions is rather simple.

We assume the well-known relational algebra operators: σ_{cond}, $\bowtie$, $\bowtie_\theta$, π, ρ (for renaming) as well as the recursive operator α [Agr87]. Utilizing the α operator is a feasible approach since most of today's modern database systems support some kind of recursive queries, e.g., Oracle with the `CONNECT BY` clause or IBM DB2 with `RECURSIVE UNION`. We use the notation:

$$\omega(Exp) \mapsto \overline{\omega}(Exp)$$

to express the transformation of an expression $\omega(Exp)$ into a relational expression $\overline{\omega}(Exp)$ by substituting ω with $\overline{\omega}$. Furthermore, *eval*(Exp) means the evaluation of the expression Exp, and *relation*(T) denotes the relation with name T.

The transformation of a query expression into relational algebra can now be specified by the following set of rules:

(1) $\mathcal{A} \mapsto$ ANNOTATION

(2) $\mathcal{C} \mapsto$ CONCEPT

(3) $\mathcal{D} \mapsto$ DOCUMENT

(4) if *eval*$(Exp) \subseteq \mathcal{A}$ then
$\sigma_P(Exp) \mapsto \sigma_{\overline{P}}(Exp) \bowtie \rho_{A_1}(\text{ANNOTATIONPVAL}) \bowtie \cdots \bowtie \rho_{A_n}(\text{ANNOTATIONPVAL})$
Here, $\overline{P}$ is constructed as follows: for each clause c_i of the form "<*prop*> <*op*> <*val*>" there is a corresponding clause "A_i.name = '<*prop*>' $\wedge$ A_i.val <*op*> <*val*>".

(5) if *eval*$(Exp) \subseteq \mathcal{D}$ then
$\sigma_P(Exp) \mapsto \sigma_P(Exp)$

(6) if *eval*$(Exp) \subseteq \mathcal{C}$ then
$\sigma_P(Exp) \mapsto \sigma_P(Exp \bowtie \text{CONCEPTTERM})$

(7) $\Phi_r(Exp) \mapsto T_{to} \leftarrow \pi_{to}(\sigma_{id=r}(\text{CONCEPTRELSHIP})$;
$\pi_{T_{to}.*}(Exp \bowtie_{id=from} (\sigma_{rel=r}(\text{RELSHIP})) \bowtie_{to=id} \textit{relation}(T_{to}))$

(8) $\Phi_r^+(Exp) \mapsto T_{to} \leftarrow \pi_{to}(\sigma_{id=r}(\text{CONCEPTRELSHIP})$;
$\pi_{T_{to}.*}(Exp \bowtie_{id=from} \alpha(\pi_{from,to}(\sigma_{rel=r}(\text{RELSHIP})) \bowtie_{to=id} \textit{relation}(T_{to}))$

Rules (1)–(3) specify simple substitutions of the sets by the corresponding relations. In rules (4)–(6), the selection operator is translated taking the normalization of the relations into account. Rule (7) translates the path traversal operator for a 3-way join between the *from* table, the *to* table and the RELSHIP table. The transformation of path expressions as part of selection conditions is handled in a very similar way and therefore omitted here. Finally, using rule (8) the path traversal operation based on the transitive closure is translated into an algebra expression that obtains all possible edges of the given relationship concept and joins them with the *from* and *to* table using the α operator.

These transformation rules are always applied in an inside-out manner. For example, for the query expression

$$\Phi_{annotates}(\Phi_{hasAnnotation}(\Phi_{ofConcept}(\Phi_{annotatedBy}(\sigma_{uri=...}(\mathcal{D})))))$$

the resulting algebra expression is shown in Fig. 4.

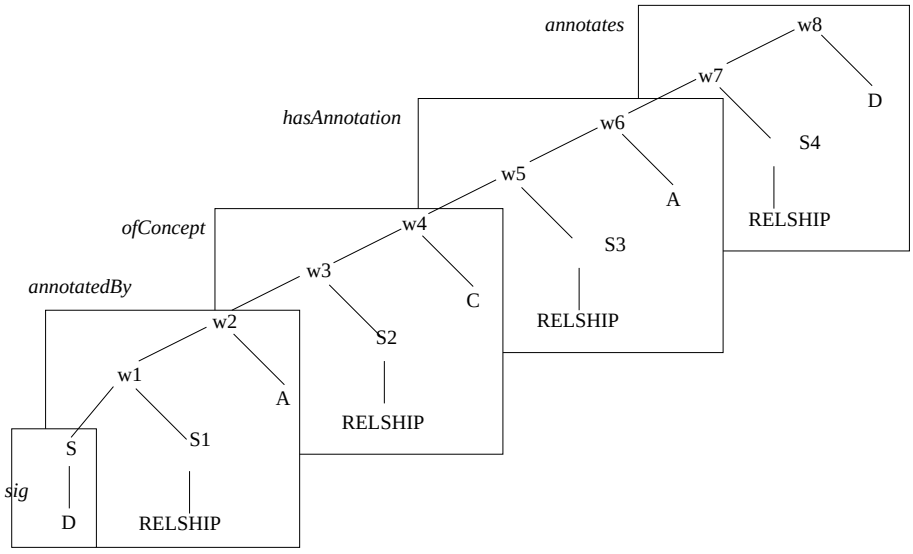

Figure 4: Transformed query tree

From this algebra expression an equivalent SQL query can be derived in a straightforward manner and therefore this step is omitted here.

6 Service Implementation

In this section we present selected services ad applications of our prototype implementation of the described annotation graph framework. We will focus mainly on the query service as the central component of the system as well as on the applications built on top of it.

In order to be able to formulate queries involving query operators introduced in Section 3, we have designed a simple language in the spirit of XPath. Here, the sets $\mathcal{A}$ (*annotation*), $\mathcal{C}$ (*concept*), and $\mathcal{D}$ (*document*) are valid root elements. If views as filters on these sets are supported, they can be used as root elements as well.

Selections are formulated by appending [*condition*] to a term. In *condition*, the properties of the objects of the set can be accessed and – in combination with the usual operators and logical connectors – used for formulating predicates. The Φ-operator is expressed by appending */relship* to the term. Here, *relship* denotes the relationship concept that has to be used for following the links. The optional + indicates that the transitive closure has to be computed.

Using these notations, the example queries given in Section 3.3 are formulated as follows:

$$\Phi_{annotates}(\Phi_{hasAnnotation}(\Phi^{+}_{isA}(\Phi_{ofConcept}(\Phi_{annotatedBy}(\sigma_{uri=\ldots}(\mathcal{D}))))))$$

document[uri=...]/annotatedBy/ofConcept/isA+/hasAnnotation/annotates

$$\Phi_{annotates}(\sigma_{ofConcept.cname=C}(\Phi_{annotatedBy}(\sigma_{uri=\ldots}(\mathcal{D}))))$$

document[uri=...]/annotatedBy[ofConcept.cname='C']/annotates

The query engine for this language is implemented as a basic Web service providing a SOAP interface [SOAP]. It accepts a query expression in the notation introduced above and translates this to a relational algebra expression according the mapping rules from Section 5. Finally, an SQL query is derived from this algebra expression and sent to the backend DBMS for evaluation. The result set is encoded into a XML document and returned to the client. A fragment of such a document containing an annotation is shown in Fig. 5. For convenience, the document contains not only the pure annotation properties but also the information about the concept of this annotation as well as information about the annotated document.

The services for creating, modifying, and deleting concepts and annotations are implemented as SOAP services, too. Loading a concept model or even a complete annotation graph is supported by an additional service which accepts an XML document containing a serialized graph of annotations and concepts. The DTD for these documents is given in Appendix A.

The query and creation services can be used by an annotation browser, for example a slightly modified version of Annozilla[3] – an annotation-capable extension to the Mozilla browser – or special browsers for annotating and viewing scientific images. Together with the document loaded into the browser, the annotations can be retrieved via SOAP from the query service and displayed in the sidebar. The improvement of our approach compared with the original Annozilla is the opportunity of navigating the graph of annotations and concepts relevant to the current document. In this way, explicit (and externally annotated) relationships between Web

[3] http://annozilla.mozdev.org

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope ...">
<SOAP-ENV:Body>
<ns1:queryResponse xmlns:ns1="urn:AnnotateIt">
<return xmlns:ns2="http://xml.apache.org/xml-soap" xsi:type="ns2:Vector">
<item xsi:type="ns1:Annotation">
<annotates xsi:type="xsd:string">http://www.docs.org/doc1.html</annotates>
<context xsi:type="xsd:string">/html/body/h1</context>
<creator xsi:type="xsd:string">Michael</creator>
<uri xsi:type="xsd:string">http://www.concepts.org/annotation#1</uri>
<ofConcept xsi:type="xsd:string">http://www.concepts.org/concept#6</ofConcept>
<created xsi:type="xsd:timeInstant">2001-11-13T00:00:00Z</created>
</item>
</return>
</ns1:queryResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5: Fragment of a XML document returned by the Query Service

documents can be followed. Furthermore, by selecting an annotation the underlying concept and as well as related annotations and documents can be explored.

Two other applications – a search engine and a catalog – are implemented directly on top of the query service as Java Server Pages (JSP). In principle, the search engine is a simple front-end to the query service: a given query is sent to the service and the results are shown with navigation links to the individual objects and their relationships.

The catalog JSP builds a Yahoo-like catalog of concepts and related (i.e., by their annotations) documents. For this purpose a root concept and a primary relationship have to be given by the user. The first page of the catalog displays the root concepts and all concepts with a direct relationship of the given type together with links to documents that are annotated by these concepts. By selecting one of the child concepts a next page is displayed containing the selected child concept as root and all concepts related to this and so on. Assuming a root concept C_R and a property relationship *relates* the following query expression obtains all directly related concepts

concept[cname='C_R']/relates

For each concept C of this set the annotated documents are obtained by

concept[cname='C']/hasAnnotation/annotates

However, because both the related concept and the document information are included in the SOAP reply document, practically all the required information for building a single catalog page can be derived using the query

concept[cname='C_R']/relates/hasAnnotation

In summary, the improvement is the flexible structuring of the catalog: The root concept and the relationships are no longer predefined. Beside the usual “is a” relationship any other kinds like temporal or spatial relationships which are also important for scientific applications can be used.

7 Conclusions

Metadata play a crucial role in integrating and linking data from heterogeneous sources, particularly if they are “conceptualized”, i.e., are underlayed by a controlled vocabulary. However, not in all scenarios metadata is embeddable in the described content: either because the metadata is not known at creation time or simply because the media of data does not support embedding or assigning additional information, e.g., like raster images or other binary data. Here, external metadata in form of data annotations offer an appropriate solution. Particularly, in research communities working with scientific data we suppose that manual annotations of data provide a usable way.

In this paper, we have presented a framework supporting such scenarios. The framework comprises a graph-based data model, operations for traversing and querying graphs of concepts and annotations, a service-oriented architecture and several domain-independent as well as domain-specific applications. Our approach is not primary intended as a general purpose solution of concept-based metadata, but we believe it shows the potential of bridging the gap between the ontology-oriented work of the Semantic Web community and approaches dedicated to data integration and linking problems.

Future issues of our work will address mainly bootstrapping and scalability. Bootstrapping tackles the problem of obtaining initial annotations based on a given set of concepts for (possibly) preexisting documents with low effort. One possible way is to establish a data registry where documents that have to be published in the Web are registered and associated with some simple annotations. Another approach particularly for text documents could be an automatic extraction of embedded metadata, e.g. based on a standard vocabulary like Dublin Core.

The scalability issue addresses the support of large communities. Due to the centralized graph storage, the presented architecture and the implemented services are appropriate only for smaller communities. A distributed approach using multiple instances of the graph storage and the query service could revoke this limitation. However, this requires replicating of essential data like concepts and partitioning of annotation and relationship sets. In addition, the effects of this data distribution on the query transformation has to be considered.

A DTD for an annotation exchange format

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT agm (concept | annotation | relationship)*>

<!--
      Concept
  -->
<!ELEMENT concept (name, definition, authority, term+, properties?,
                   isA*, roles?)>
<!-- the URI that uniquely identifies the concept
     of the form: urn:agm/concept#<id> (<id> is an integer)
 -->
<!ATTLIST concept id CDATA #REQUIRED>

<!-- the name (primary term) of the concept -->
<!ELEMENT name (#PCDATA)>

<!-- the definition in form of an explaining text of
     the concept -->
<!ELEMENT definition (#PCDATA)>

<!-- the authority of the concept, i.e., the person responsible
     for the concept -->
<!ELEMENT authority (#PCDATA)>

<!-- a term (i.e., a synonym) for the concept -->
<!ELEMENT term (#PCDATA)>

<!-- a set of property definitions -->
<!ELEMENT properties (property+)>
<!ELEMENT property EMPTY>
<!ATTLIST property name CDATA #REQUIRED
                   kind CDATA #REQUIRED>

<!-- a set of parent concepts -->
<!ELEMENT isA EMPTY>
<!ATTLIST isA idref CDATA #REQUIRED>

<!-- if the concept is used for defining relationships,
     these elements describe the related classes (realm)
     an assign role names -->
<!ELEMENT roles (fromRole, toRole)>

<!ELEMENT fromRole (#PCDATA)>
<!ATTLIST fromRole realm CDATA #IMPLIED >
```

```
<!ELEMENT toRole (#PCDATA)>
<!ATTLIST toRole realm CDATA #IMPLIED >

<!--
    Annotation
  -->
<!ELEMENT annotation (creator, created, annotates, values?)>
<!ATTLIST annotation id        CDATA    #REQUIRED
                     ofConcept CDATA #REQUIRED>

<!-- the author of the annotation -->
<!ELEMENT creator (#PCDATA)>

<!-- the creation time -->
<!ELEMENT created (#PCDATA)>

<!-- a set of property instantiations -->
<!ELEMENT values (pvalue+)>

<!ELEMENT pvalue EMPTY>
<!ATTLIST pvalue name  CDATA #REQUIRED
                 kind  CDATA #REQUIRED
                 value CDATA #REQUIRED>

<!-- a representation of the "annotates" relationship -->
<!ELEMENT annotates (target, context?)>
<!ELEMENT target EMPTY>
<!ATTLIST target url CDATA #REQUIRED>

<!ELEMENT context (#PCDATA)>

<!--
    Relationship
  -->
<!ELEMENT relationship (from, to, values?)>
<!ATTLIST relationship ofConcept CDATA #REQUIRED>

<!-- the objects (concepts, annotations etc.) participating on the
     relationship -->
<!ELEMENT from EMPTY>
<!ATTLIST from idref CDATA #REQUIRED>

<!ELEMENT to EMPTY>
<!ATTLIST to idref CDATA #REQUIRED>
```

References

[Agr87] R. Agrawal: Alpha: An Extension of Relational Algebra to Express a Class of Recursive Queries. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, 580–590, ACM, 1987.

[BE96] O. A. Bukhres, A. K. Elmagarmid (eds.): *Object-Oriented Multidatabase Systems - A Solution for Advanced Applications*. Prentice Hall, Hempel Hempstead, 1996.

[BFM+97] C. Baru, R. Frost, R. Marciano, R. Moore, A. Rajasekar, M. Wan: Metadata to Support Information Based Environments. In *Proc. of the 2nd IEEE Metadata Conference*, IEEE, 1997.

[BG01] J.-M. Bremer, M. Gertz: Web Data Indexing Through External Semantic-carrying Annotations. In *Eleventh International Workshop on Research Issues in Data Engineering (RIDE'01)*, 69–76, IEEE, 2001.

[CD99] J. Clark, S. DeRose. XML Path Language (XPath) Version 1.0, W3C Recommendation, Nov 1999.

[COS98] K.E. Campbell, D.E. Oliver, E.H. Shortliffe: The unified medical language system: towards a collaborative approach for solving terminology problems. *Journal of the American Medical Informatics Association*, Volume 8, 12–16, 1998.

[DC0] Dublin Core Metadata Initiative, dublincore.org/.

[DEFS99] S. Decker, M. Erdmann, D. Fensel, R. Studer: Ontobroker: Ontology based Access to Distributed and Semi-Structured Information. In *Database Semantics - Semantic Issues in Multimedia Systems, IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics (DS-8)*, 351–369. Kluwer, 1999.

[ERS99] A. Elmagarmid, M. Rusinkiewicz, A. Sheth (eds.): *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann Publishers, San Francisco, 1999.

[FAD+99] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, S. Staab, R. Studer, A. Witt. On2broker: Semantic-based access to information sources at the WWW, 1999. In: Proceedings of the World Conference on the WWW and Internet (WebNet 99), 1999.

[FFR96] A. Farquhar, R. Fikes, J. Rice: The Ontolingua Server: A Tool for Collaborative Ontology Construction. Technical Report KSL-96-26, Knowledge Systems Laboratory, Stanford, CA, 1996.

[Gar99] J. Garfunkel: Web Annotation Technologies. look.boston.ma.us/garf/webdev/annote/software.html

[Gru93] T. R. Gruber: Toward Principles for the Design of Ontologies user for Knowledge Sharing. *International Journal on Human-Computer Studies (1993)*.

[HH00] J. Heflin, J. Hendler: Dynamic Ontologies on the Web. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, 443–449, AAAI/MIT Press, 2000.

[HLO99] R.M. Heck, S.M. Luebke, C.H. Obermark: A Survey of Web Annotation Systems, www.math.grin.edu/∼luebke/Research/Summer1999/survey_paper.html

[Ino] Inote: An Image Annotation Tool in Java, jefferson.village.virginia.edu/inote/.

[ISO99] ISO/IEC 13250 Topic Maps. www.y12.doe.gov/sgml/sc34/document/0129.pdf.

[Kar96] P. D. Karp: A Strategy for Database Interoperation. *Journal of Computational Biology 2:4 (1996)*, 573–586.

[KH97] S. Koslow, M. Huerta (eds.): *Neuroinformatics: An Overview of the Human Brain Project*. Lawrence Erlbaum Associates, NJ, 1997.

[KKHS01] J. Kahan, M.-R. Koivunen, E. P. Hommeaux, R. R. Swick: Annotea: An Open RDF Infrastructure for Shared Web Annotations. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*, 623–632, ACM, 2001.

[KS98] V. Kashyap, A. Sheth: Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies, In *Cooperative Information Systems: Current Trends and Directions*, 139–178, Academic Press, 1998.

[LS99] O. Lassila, R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification, 1999. W3C Recommendation, February, 1999.

[MPE98] R. Moore, T. Prince, M. Ellisman: Data-Intensive Computing and Digital Libraries. *CACM 41:11 (November 1998)*, 56–62.

[NIH] Neuroinformatics – The Human Brain Project. www.nimh.nih.gov/neuroinformatics/index.cfm.

[Oli98] D.E. Oliver: Synchronization of Diverging Versions of a Controlled Medical Terminology. In *Proceedings of the 1998 AMIA Annual Fall Symposium*, 850–854, 1998.

[OSS99] D.E. Oliver, Y. Shahar, E.H. Shortliffe, M.A. Musen: Representation of changes in controlled medical terminologies. *Artificial Intelligence in Medicine*, Volume 15, 53–76, 1999.

[Pep00] S. Pepper: The TAO of Topic Maps. www.gca.org/papers/xmleurope2000/papers/s11-01.html.

[PW97] T. A. Phelps, R. Wilensky: Multivalent Annotations. In *Research and Advanced Technology for Digital Libraries – First European Conference, ECDL'97*, 287–303, LNCS 1324, Springer-Verlag, Berlin, 1997.

[She99] A. P. Sheth: *Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics*. Kluwer Academic Press, 1999.

[SOAP] Simple Object Access Protocol (SOAP) 1.1. W3C Note, May 2000.

[SPKR96] B. Swartout, R. Patil, K. Knight, T. Russ. Toward Distributed Use of Large-Scale Ontologies, 1996. In Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, Alberta, Canada, 1996.

[TRW+97] K. Tochtermann, W.-F. Riekert, G. Wiest, J. Seggelke, B. Mohaupt-Jahr: Using Semantic, Geographical, and Temporal Relationships to Enhance Search and Retrieval In Digital Catalogs. In *Research and Advanced Technology for Digital Libraries – First European Conference, ECDL'97*, 73–86, LNCS 1324, Springer-Verlag, Berlin, 1997.

[UCD] UC Davis/UC San Diego Human Brain Project Informatics of the Human and Monkey Brain, neuroscience.ucdavis.edu/HBP/

[W3C0] World Wide Web Consortium: Annotea Project, www.w3.org/2001/Annotea/.

[Wie92] G. Wiederhold: Mediators in the Architecture of Future Information Systems. *IEEE Computer 25:3 (March 1992)*, 38–49.

[WMG+99] R. Williams, P. Messina, F. Gagliardi, J. Darlington, G. Aloisio (eds.): *European–United States Joint Workshop on Large Scientific Databases*. Center for Advanced Computing Research, California Institute of Technology, Pasadena, CA, September 1999.