

Data Annotation Models

Neerja Bhatnagar
Computer Science
Department
California State University,
Chico
Chico, CA 95929-0003
neerja@cs.ucsc.edu

Ben A. Juliano
Computer Science
Department
California State University,
Chico
Chico, CA 95929-0003
juliano@csuchico.edu

Renee S. Renner
Computer Science
Department
California State University,
Chico
Chico, CA 95929-0003
renner@csuchico.edu

ABSTRACT

This paper addresses the problem of the unsuitability of most relational databases to express *data annotations* by defining data annotation models that allow users to annotate relational data at five levels - database, relation, column, tuple, and cell. These models are structured using Extensible Markup Language (XML). The most important feature of these models is that they do not require any structural or schematic changes to the underlying database. It is common knowledge that database administrators (DBAs) are very resistant to making any structural or schematic changes to their deployed databases. Moreover, these models are simple to understand, flexible, extensible, customizable, database-neutral, and platform-independent. These models also allow users to cross-reference related annotations. Simplicity and the ease of understanding is the main motivation for the design of the data annotation models presented in this paper.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous;
D.2.8 [Information Systems]: Design for data annotation models to annotate relational data.

General Terms

semantic data annotations

Keywords

annotating relational data, data annotations, semantic annotations

1. INTRODUCTION

The metadata schema in most relational databases is unsuitable for expressing *data annotations*. Data annotations are defined as semantically rich metadata applicable to a particular application domain that help further clarify *features of interest*.

A feature of interest is a data item that a user wants to annotate [8]. Types of data annotations include comments, descriptions, definitions, notes, error messages, among several others. Most relational databases utilize their metadata schema to store statistical information for constraint checking and query optimization.

This paper presents data annotation models that address the problem of the unsuitability of metadata schema provided by relational databases to express data annotations. These models allow users to annotate relational data at five levels of granularity - database, relation, column, tuple, and cell. The most important characteristic of these models is that they do not require any structural or schematic changes to the underlying database. It is common knowledge that database administrators (DBAs) are resistant to structural and schematic changes to their already deployed databases. The data annotation documents based on these models reside, outside the database, on the host file system of a computer system. Thus, these models stand a greater chance of being adopted for existing database deployments. In addition, these models are easy to understand, flexible, extensible, customizable, database-neutral, platform-independent, and allow users to cross-reference related annotations. The ability to customize these data annotation model stems from the flexibility given to users to declare an annotation based on their individual requirements. Specifically, users can customize the `applicationDomainSpecificTag` tag according to their specific requirements. Simplicity and ease of understanding are the key motivating factors for the development of the data annotation models presented in this paper.

Data annotations reduce communication and data exchange hassles and provide almost all database users (scientists, customer service providers, banks, corporations) with a more collaborative environment. A scientist, who wants to share the discovery he or she made while investigating an image can utilize annotations to annotate the image. He or she can also seamlessly share these findings with other researchers. Due to its numeric nature, it is often difficult to interpret the semantics of scientific data, simply by looking at it. As an example, it is difficult to interpret whether the data in the `TEMPERATURE` column is expressed in Metric or English units. Annotations can help scientists to annotate the `TEMPERATURE` column with the appropriate unit. If the column contains temperature in both Metric and English units, the cell-level data annotation model can be used to annotate each cell individually with its unit. An alternative is to change the data type of the `TEMPERATURE` column, from `DECIMAL` to `VARCHAR`, to accommodate the unit. This might

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

not be desirable since the scientists will lose the ability to manipulate the data mathematically. Moreover, this also requires a change to the underlying RDBMS. In some cases, annotations might also help reduce costs, and save time and effort. As an example, a customer can dispute a charge on his or her bank statement using annotations. The customer need not restrict himself or herself to the customer service hours. A customer service provider can also use annotations to update the customer.

The rest of this paper is organized as follows - Section 2 compares and contrasts the data annotation models presented in this paper with Annotea, DBNotes, MONDRIAN, SLIMPad and those presented in [8] and [6]; Section 3 presents the data annotation models that allow users to annotate relational data at five different levels of granularity - database, relation, column, tuple, and cell; Section 4 discusses a few ways of extending the work presented in this paper; and Section 5 presents conclusions and future work.

2. RELATED WORK

This section compares and contrasts the annotation models presented in this paper with Annotea [9], DBNotes [5], MONDRIAN [7], SLIMPad [10], and with those presented in [8] and [6]. Annotea allows users to annotate documents identified by a URI with or without the knowledge of the authors. Similar to the models presented in this paper, DBNotes and MONDRIAN annotate relational data. The models presented in this paper deal with relational data to be annotated at five different levels - database, relation, column, tuple, and cell. DBNotes focuses on the where-provenance and annotation propagation. MONDRIAN focuses on biological databases, and allows users to annotate both single values and the association between multiple values. SLIMPad annotates data that resides in a variety of applications, such as databases, spreadsheets, and documents, among several others. SLIMPad addresses annotations in the domain of physicians providing treatment to patients. The system presented in [6] annotates audio-visual documents. The system presented in [8] annotates neuroanatomical images at various levels of granularity.

Annotea and SLIMPad utilize RDF to define annotations. DBNotes and MONDRIAN utilize relational data to express annotations. The annotation system in [6] utilizes LEDA graph structure and XML Document Object Model (DOM) to express annotations. The data annotation models presented in this system also utilizes XML to structure data annotations. The system presented in [8] uses text-based annotations. All annotations, by default, based on the data annotation models presented in this paper, are accompanied with metadata information, such as the author's name and the creation time stamp. This behavior is common with annotations in Annotea. Both Annotea and the data annotation models presented in this paper allow annotations to cross-reference related annotations.

Annotations in Annotea reside in generic RDF databases, accessible through HTTP servers. Annotations presented in [8], [6], and in this paper reside outside the underlying database whose data they annotate. Data annotation documents based on the data annotation models presented in this paper reside on the file system of a computer system. SLIMPad modifies the base layer that it annotates in order to store annotations. DBNotes and MONDRIAN store annotations within the RDBMS. Columns may be added to relations in

```
<annotationDocument>
<documentName>uniqueName</documentName>
<documentId>uniqueId</documentId>
<annotationAttachedTo>
  <database>databaseName</database>
</annotationAttachedTo>
<!-- annotation and annotationMetadata may occur
multiple times in a single document -->
<annotation>
  <applicationDomainSpecificMetatag>
    dataAnnotation
  </applicationDomainSpecificMetatag>
  <annotationMetadata>
    <author>someone</author>
    <recorded>someDateAndTime</recorded>
  </annotationMetadata>
</annotation>
<referencedAnnotations>
  <documentNameList>
    <documentName>uniqueName</documentName>
    <documentName>uniqueName</documentName>
    ...
  </documentNameList>
</referencedAnnotations>
</annotationDocument>
```

Figure 1: Database-Level Data Annotation Model

order to annotate relational data [3]. Annotations at column and tuples levels can be expressed using this scheme. However, it is difficult to express annotations at the database, relation, and cell levels.

3. DATA ANNOTATION MODELS

Figures 1-5 present data annotation models that allow users to annotate relational data at five different levels of granularity - database, relation, column, tuple, and cell. XML is used to structure these data annotation models. XML was chosen because it provides several advantages over other data formats, such as text, e-mail, or electronic forms. XML is database-neutral and platform-independent. XML's database-neutrality allows the use of same data annotation models to express data annotations on base data that resides in heterogeneous databases, such as, DB2 UDB, Oracle, SQL Server etc. XML's platform-independence allows users to express, share, and utilize data annotations irrespective of the operating system and platform they use. Since XML supports Unicode, the support for expressing data annotations in several different languages is already built-in. Therefore, it can be used to share data seamlessly regardless of the underlying database and operating system. XML supports Unicode, and therefore, can support several different languages [11].

Modules. Each of the data annotation model presented in Figures 1-5 may be further divided into modules, namely, identification, level, annotation, annotation metadata, and cross-reference. These modules are illustrated in Fig. 6. The identification module uniquely identifies a data annotation document. It contains the hierarchy within the `documentName` and `documentId` tags. The level module represents the level (database, relation, column, tuple, or cell) that a data annotation document annotates. The level module is identified by the tag `annotationAttachedTo`. The annotation module contains the actual data annotation, and the annotation metadata module maintains bookkeeping information (creation time stamp and author) on the actual annotation. The annotation module is characterized by the tag `applicationDomainSpecificTag`, which is enclosed within the `annotation` tag. The annotation metadata module contains the node hierarchy within the `annotationMetadata`

```

<annotationDocument>
  <documentName>uniqueName</documentName>
  <documentId>uniqueId</documentId>
  <annotationAttachedTo>
    <database>databaseName</database>
    <relation>relationName</relation>
  </annotationAttachedTo>
  <!--annotation and annotationMetadata may occur
multiple times in a single document -->
  <annotation>
    <applicationDomainSpecificMetatag>
      dataAnnotation
    </applicationDomainSpecificMetatag>
    <annotationMetadata>
      <author>someone</author>
      <recorded>someDateAndTime</recorded>
    </annotationMetadata>
  </annotation>
  <referencedAnnotations>
    <documentNameList>
      <documentName>uniqueName</documentName>
      ...
    </documentNameList>
  </referencedAnnotations>
</annotationDocument>

```

Figure 2: Relation-Level Data Annotation Model

```

<annotationDocument>
  <documentName>uniqueName</documentName>
  <documentId>uniqueId</documentId>
  <annotationAttachedTo>
    <database>databaseName</database>
    <relation>relationName</relation>
    <column>columnName</column>
    <!--for composite primary keys, list
values separated by commas-->
    <tuple>primary key</tuple>
  </annotationAttachedTo>
  <!--annotation and annotationMetadata may occur
multiple times in a single document -->
  <annotation>
    <applicationDomainSpecificMetatag>
      dataAnnotation
    </applicationDomainSpecificMetatag>
    <annotationMetadata>
      <author>someone</author>
      <recorded>someDateAndTime</recorded>
    </annotationMetadata>
  </annotation>
  <referencedAnnotations>
    <documentNameList>
      <documentName>uniqueName</documentName>
      ...
    </documentNameList>
  </referencedAnnotations>
</annotationDocument>

```

Figure 5: Cell-Level Data Annotation Model

```

<annotationDocument>
  <documentName>uniqueName</documentName>
  <documentId>uniqueId</documentId>
  <annotationAttachedTo>
    <database>databaseName</database>
    <relation>relationName</relation>
    <column>columnName</column>
  </annotationAttachedTo>
  <!--annotation and annotationMetadata may occur
multiple times in a single document -->
  <annotation>
    <applicationDomainSpecificMetatag>
      dataAnnotation
    </applicationDomainSpecificMetatag>
    <annotationMetadata>
      <author>someone</author>
      <recorded>someDateAndTime</recorded>
    </annotationMetadata>
  </annotation>
  <referencedAnnotations>
    <documentNameList>
      <documentName>uniqueName</documentName>
      ...
    </documentNameList>
  </referencedAnnotations>
</annotationDocument>

```

Figure 3: Column-Level Data Annotation Model

```

<annotationDocument>
  <documentName>...</documentName> ← Identification
  <documentId>...</documentId> ← Identification
  <annotationAttachedTo>
    <database>...</database> ← Level
  </annotationAttachedTo>
  <annotation>
    <applicationDomainSpecificMetatag>
      dataAnnotation ← Annotation
    </applicationDomainSpecificMetatag>
    <annotationMetadata>
      <author>...</author> ← Annotation Metadata
      <recorded>...</recorded>
    </annotationMetadata>
  </annotation>
  <referencedAnnotations> ← Cross-Reference
    <documentNameList>
      <documentName>...</documentName>
    </documentNameList>
  </referencedAnnotations>
</annotationDocument>

```

Figure 6: Modules

```

<annotationDocument>
  <documentName>uniqueName</documentName>
  <documentId>uniqueId</documentId>
  <annotationAttachedTo>
    <database>databaseName</database>
    <relation>relationName</relation>
    <!--for composite primary keys, list values separated
by commas-->
    <tuple>primary key</tuple>
  </annotationAttachedTo>
  <!--annotation and annotationMetadata may occur multiple
times in a single document -->
  <annotation>
    <applicationDomainSpecificMetatag>
      dataAnnotation
    </applicationDomainSpecificMetatag>
    <annotationMetadata>
      <author>someone</author>
      <recorded>someDateAndTime</recorded>
    </annotationMetadata>
  </annotation>
  <referencedAnnotations>
    <documentNameList>
      <documentName>uniqueName</documentName>
      ...
    </documentNameList>
  </referencedAnnotations>
</annotationDocument>

```

Figure 4: Tuple-Level Data Annotation Model

tag. By default, each annotation is accompanied with the annotation metadata, namely the name of the author of the annotation and the creation time stamp of the annotation. The cross-reference module (characterized by the element hierarchy within the `referencedAnnotations` tag) allows related annotations to cross-reference each other. Annotations are, in essence, immutable i.e. an annotation that semantically overrides another annotation does not cause the original annotation to be erased.

Storage. Data annotation documents based on the models presented above reside on the file system provided by a computer system. When the number of data annotation documents becomes prohibitively large, a smart indexing scheme would become necessary to quickly locate and retrieve annotations. An alternative is to store annotation documents on disks that are being designed specifically to store and efficiently retrieve semi-structured data [2]. Annotation documents may also be stored inside a relational database. XML documents stored in their entirety inside relational databases present a few problems. The first problem is that the storage of annotation documents inside the

relational database that contains the base data being annotated would require structural and schematic changes to the database. The second problem is that all queries that use “select *” might need to be modified. Similarly, “insert into” queries that do not mention the column names explicitly would need to be modified. Thirdly, an SQL-XML parser would be needed to process queries that pertain to annotation documents. XML documents can be shredded and parsed in order to convert them into tabular format suitable for mapping on to relational data [1]. This defeats the whole concept of semi-structured data format - the data to begin with is not suitable for storage into a relational database. Moreover, putting shredded XML documents back together is expensive since it requires the computation of several joins. Annotation documents can also be stored in XML native databases. However, retrieving these annotations from XML native databases requires the knowledge of XQuery. XQuery is a complex query language, and might present users with a steep learning curve.

Alternatives and their Disadvantages. An alternative to using the data annotation models presented in Figures 1-5 is to keep notes in text documents. However, this approach presents a few problems. First, this alternative does not provide a uniform, consistent mechanism to express annotations. The reason is that each user would use his or her own personal format. Second, sharing text documents across platforms is difficult. Third, keeping annotations in text documents does not automatically maintain metadata information on annotations. The data annotation models presented in this paper not only address all of these problems, but also present useful features, such as, cross-referencing and annotation metadata.

Another alternative to using the data annotation models presented in Figures 1-5 is to declare ANNOTATION columns. An ANNOTATION column may be added for each data column. A single ANNOTATION column can express tuple-level annotations. However, this technique requires the addition of columns to an already deployed database. This might be problematic because it is common knowledge that DBAs are resistant to making any changes to already deployed databases. Secondly, this technique is only effective for annotating data at the column and tuple levels. The third problem with this technique is that all queries with “select *” must be modified to exclude ANNOTATION columns, particularly, if the users want to view data only. Similarly, all data insertion queries that use “insert into relation values()” without specifying the columns into which data has to be inserted must also be modified. All applications that retrieve data from the database, and manipulate and process this data must be modified accordingly to accommodate the changes in the underlying database. Such changes to the underlying database, and applications that work in association with the database, can prove to be particularly difficult for production systems that are typically heavily used. Another major disadvantage of this technique is that it might not be possible to cross-reference related annotations since the annotations stored in database columns are no longer uniquely distinguishable.

Although, it might be feasible to utilize ANNOTATION columns to annotate relational data at the column and tuple level, this technique cannot be used effectively to annotate at the database, relation, and cell levels. This is because it is hard to decide which relation should host the column that anno-

```
<annotationDocument>
  <documentName>RE1</documentName>
  <documentId>RE1</documentId>
  <annotationAttachedTo>
    <database>REAL_ESTATE</database>
  </annotationAttachedTo>
  <annotation>
    <actionRequired>
      Please create a separate database for properties
      listed at $1,000,000 or more. Separate out real
      estate agents that deal exclusively in these
      properties. Proposed database name - LUXURY_ESTATES.
      Name the relations using firm's standard naming
      conventions.
    </actionRequired>
    <annotationMetadata>
      <author>TheManager</author>
      <recorded>April 2, 2004 10:37:15 PM</recorded>
    </annotationMetadata>
  </annotation>
</annotationDocument>
```

Figure 7: Database-Level Example Annotation Document

```
<annotationDocument>
  <documentName>plntlogyLateTriassic1</documentName>
  <documentId>Pal1</documentId>
  <annotationAttachedTo>
    <database>PALEONTOLOGY</database>
    <relation>LATE_TRIASSIC</relation>
  </annotationAttachedTo>
  <annotation>
    <description>
      Neither flowering plants nor grass existed. The
      ground was covered with ferns and mosses. Plant life
      was very drab - just green and brown in color.
    </description>
    <annotationMetadata>
      <author>paleontologist</author>
      <recorded>Apr 17, 2004 8:02:08 PM</recorded>
    </annotationMetadata>
  </annotation>
</annotationDocument>
```

Figure 8: Relation-Level Example Annotation Document

tates the entire database. Similarly a column must be added to each relation to store annotations at the relation and cell levels.

Examples. Fig. 7 presents an example database-level data annotation document that a real estate firm’s manager might use to convey to his or her DBA to transfer the records of properties for over a million dollars into a separate database. With the use of data annotations, the manager does not need to schedule a meeting with the DBA. The DBA can carefully review his or her manager’s annotation, make the appropriate changes, and inform the manager after transferring the pertinent data into a separate database.

Fig. 8 presents an example data annotation document at the relation level that a paleontologist might use to annotate a relation in a paleontology database with information on topography or habitat existent during a particular period. This document annotates the *LateTriassic* relation. The *LateTriassic* relation may contain information on *Coelophysis*, *Cynodont*, and *Placerias*, among others. However, these relations would not contain information on the state of the earth, or the habitat, or the weather during the *LateTriassic* period.

Fig. 10 illustrates how a chef can utilize the column-level data annotation model (presented in Fig. 3) to annotate the *Quantity* column in the relation *CondimentList* (see Fig. 9). Fig. 11 presents an example cell-level data annotation document that a chef can use to annotate the

Id	Name	Quantity
1	Salt	3
8	Turmeric	2

Figure 9: Relation CondimentList

```
<annotationDocument>
<documentName>cookingRecipeQuantity1</documentName>
<documentId>cooking1</documentId>
<annotationAttachedTo>
<database>COOKING</database>
<relation>RECIPE</relation>
<column>QUANTITY</column>
</annotationAttachedTo>
<annotation>
<note> All quantities in tablespoons.
Conversion: 1 tablespoon = 5 milligrams </note>
<annotationMetadata>
<author>Chef Alice</author>
<recorded>May 27, 2004 9:12:24 AM</recorded>
</annotationMetadata>
</annotation>
</annotationDocument>
```

Figure 10: Column-Level Example Annotation Document

cell that contains the data value **Turmeric** in the relation **CondimentList**.

Fig. 13 presents an example data annotation document at the tuple level. The department secretary uses the tuple-level data annotation model presented in Fig. 4 to inform the graduate coordinator that John Doe has accepted their offer of admission, and therefore, his record must be moved from relation **Offered** to **Accepted** (see Fig. 12). For more details on data annotation models, please refer to [4].

4. EXTENDED DATA MODELS

As the number of data annotation documents grows, a smart indexing scheme will become necessary to locate and retrieve data annotation documents efficiently. A new storage scheme especially customized for storage and efficient retrieval of semi-structured data is also a viable option [2]. A query language, named Annotation Query Language (AnQL) may be used to query data annotation documents based on the data annotation models presented in this paper. AnQL is an SQL-like query language that is designed to make use of

```
<annotationDocument>
<documentName>cookingCondimentList</documentName>
<documentId>cooking2</documentId>
<annotationAttachedTo>
<database>COOKING</database>
<relation>CONDIMENT_LIST</relation>
<column>CONDIMENT_NAME</column>
<tuple>8</tuple>
</annotationAttachedTo>
<annotation>
<caution>
Be very careful with turmeric. Turmeric leaves
yellow stains on everything incl. counter tops,
and clothes. These stains are extremely difficult
to remove.
</caution>
<annotationMetadata>
<author>alice</author>
<recorded>Apr 27, 2004 4:18:16 AM</recorded>
</annotationMetadata>
</annotation>
</annotationDocument>
```

Figure 11: Cell-Level Example Annotation Document

NEW_STUDENT_INFO		
STUDENT_ID	NAME	DOB
999888777	John Doe	1/1/1984
123456789	Jane Smith	1/2/1986

ADMIT_INFO			
STUDENT_ID	YEAR	DEGREE	PROGRAM
999888777	2004	B.S.	ELEC. ENGG.
123456789	2004	M.S.	COMP. SCI.

Figure 12: Admission Database

```
<annotationDocument>
<documentName>admissionsNewStudentInfo
</documentName>
<documentId>adml1</documentId>
<annotationAttachedTo>
<database>ADMISSIONS</database>
<relation>NEW_STUDENT_INFO</relation>
<tuple>999888777</tuple>
</annotationAttachedTo>
<annotation>
<comment>
John Doe (Id 999888777) accepted offer.
Please change status.
</comment>
<annotationMetadata>
<author>departmentSecretary</author>
<recorded>May 27, 2004 4:03:08PM
</recorded>
</annotationMetadata>
</annotation>
</annotationDocument>
```

Figure 13: Tuple-Level Example Annotation Document

the already existent abundant SQL knowledge and skill set. With SQL-like semantics, AnQL would not present as steep a learning curve as that presented by XQuery and XPath. For more details on AnQL, please refer to [4]. Data models may also be extended to annotate data based on other models, such as object-oriented or hierarchical. A data annotation management system, designed as a virtualization over the underlying relational database, that implements both the annotation models presented above and AnQL may also be developed.

5. CONCLUSIONS AND FUTURE WORK

This paper presents data annotation models that allow users to annotate relational data at five levels of granularity of relational data - database, relation, column, tuple, and cell. These models address the problem of unsuitability of metadata schema provided by a majority of relational databases to express data annotations. These models do not require any structural or schematic changes to be made to the underlying relational database. It is common knowledge that DBAs are highly resistant to making any structural and schematic changes to their deployed databases. Therefore, these models have a better chance of getting accepted for existing database deployments. Additionally, the data annotation models presented in this paper are simple, flexible, extensible, database-neutral, and platform-independent. The documents based on these models reside outside the underlying database on the file system of computer system. A data annotation management system, that utilizes AnQL to query data annotations, may be developed to implement the data annotation models presented in this paper.

6. REFERENCES

- [1] A. H. Al-Azzawe. IBM video online for e-business -

DB2 inbound XML data fragments.

<http://www-106.ibm.com/developerworks/db2/library>,
June 2004.

- [2] M. Bhadkamkar, V. Hristidis, and R. Rangaswami. Efficient native XML storage. Technical report, Florida International University, April 2005.
- [3] D. Bhagwat, L. Chiticariu, W.-C. Tan, and G. Vijayvargiya. An annotation management system for relational databases. *VLDB*, 2004.
- [4] N. Bhatnagar. Data annotation models and annotation query language. Master's thesis, California State University, Chico, May 2006.
- [5] L. Chiticariu, W.-C. Tan, and G. Vijayvargiya. DBNotes: a post-it system for relational databases based on provenance. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 942–944, New York, NY, USA, 2005. ACM Press.
- [6] E. Egyed-Szigmond, Y. Pri, A. Mille, and J. Pinon. A graph-based audiovisual document annotation and browsing system. In *RIAO (CAIR)*, April 2000.
- [7] F. Geerts, A. Kementsietsidis, and D. Milano. Mondrian: Annotating and querying databases through colors and blocks. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, page 82, Washington, DC, USA, 2006. IEEE Computer Society.
- [8] M. Gertz, K.-U. Sattler, F. Gorin, M. Hogarth, and J. Stone. Annotating scientific images: A concept-based approach. In *SSDBM '02: Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, pages 59–68, Washington, DC, USA, 2002. IEEE Computer Society.
- [9] J. Kahan and M.-R. Koivunen. Annotea: an open rdf infrastructure for shared web annotations. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 623–632, New York, NY, USA, 2001. ACM Press.
- [10] D. Lois, M. David, B. Shawn, D. Longxing, W. Mathew, G. Paul, A. Joan, L. Mary, and L. J. A. Bundles in captivity: An application of superimposed information. Technical report, 2000.
- [11] K. B. Sall. *XML Family of Specifications A Practical Guide*. Addison Wesley, Boston MA, 2002.