The Use of Ontologies for Representing Database Schemas of Fuzzy Information

Ignacio J. Blanco,[1,†] M. Amparo Vila,[1,‡] Carmen Martinez-Cruz[2,*]
[1]*Department of Computer Science and Artificial Intelligence, University of Granada, C/Periodista Daniel Saucedo Aranda S/N, 18071, Granada, Spain*
[2]*Department of Computer Science, 035-A3, University of Jaen, Las Lagunillas Campus, 23071, Jaen, Spain*

In this paper, an ontology system is proposed to represent the knowledge structure enabling fuzzy information to be stored in fuzzy databases. This proposal allows users or applications to simplify the metadata definition process that is necessary for representing and managing imprecise and classic information in these databases. This ontology then acts as an interface that formalizes the representation of such structures and allows access to them. The instances obtained from this ontology represent the schemas that describe domain information in a database. The description of fuzzy and classic database schemas allows access to online public databases for which no other semantic description is associated. This paper also presents another ontology to represent these schemas as instances. Not only does this ontology allow fuzzy data values to be stored (because of the definition of fuzzy data types as classes of the ontology) but it also enables schema tables and attributes to be defined.

1. INTRODUCTION

Although the area of knowledge management has studied many mechanisms for representing and managing knowledge about a certain domain or application, the ability to represent knowledge with imprecision in its definition makes this process more flexible and closer to the user's natural language. Fuzzy logic was introduced by Zadeh[1,2] and establishes a frame in which information can be defined and managed using a certain level of imprecision or certainty. Since then, many knowledge representation techniques have included such capabilities. Some logical languages have been extended by the inclusion of fuzzy predicates, e.g., the Fuzzy Prolog,[3] whereas other frame-based systems (e.g., object-oriented models) have been extended by relaxing their relations and attributes,[4] and relational systems have been extended in order to manage different degrees of certainty about the stored data.[5–8]

*Author to whom all correspondence should be addressed: e-mail: cmcruz@ujaen.es.
†e-mail: iblanco@decsai.ugr.es.
‡e-mail: vila@decsai.ugr.es.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 23, 419–445 (2008)
 Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.20274

Fuzzy information lets users to express information about a certain domain as they do in natural language. Fuzzy systems mainly use linguistic labels, preference labels, or valuable-based structures to describe the environmental features that users must describe. Such fuzzy representation mechanisms enable users to store concepts such as *A person has light brown hair or blonde to brown hair*. You can also say *Jane is very tall* instead of specifying a precise height.[9]

A great number of proposals have been developed to manage fuzzy information, mostly in the area of database research,[5,6,10–13] and these extend database management systems (DBMS) so that this kind of information may be managed. These extensions range from adding a membership degree in a tuple (as proposed by Baldwin and Zhou[14] and Raju and Majumdar[15]) to using possible distributions and resemblance relations in the relational databases simultaneously (as in Rundensteiner et al.,[16] Chen Vandenbulcke and Kerre[17]). Medina et al.[12] attempt to integrate all the proposals into a single model that includes the metadata definition of fuzzy data, a definition and managing language (FSQL), and a robust implementation.[7]

Access to fuzzy information represented in fuzzy DBMS requires deep knowledge about how the information is defined in such a system. This knowledge (henceforth the *fuzzy metaknowledge base* (FMB)) enables information to be stored in the extended database (as the catalog does with classic DBs) and any user or application to access it. When new functionalities are added to exploit this kind of information such as deductive functions or data-mining operations; however, new datatypes and requirements extend the fuzzy metadata catalog and make it too complex to work with directly from the database. A new, semantically rich representation, which is closer to the user, is therefore needed to solve this problem and ontologies appear to be a good solution for this.

Many public databases (DB) are currently freely available (on Internet or via other means), and schemas of these databases are published for accessing them. Fuzzy databases also represent the semantics of an information domain (like all relational DBs), but the extensions made to the relational model must be explicitly described. There are Web applications (e.g. *ISQLPlus* ©), which are front-end databases that allow Internet access to any DB. Database schemas (fuzzy or classic) then become a complementary resource in the semantic Web that should be available. In this paper, we present a proposal for achieving this goal.

Ontologies are currently the most popular knowledge representation technique. Although they allow knowledge about a certain domain or process to be described semantically, this representation should be formal, agreed by consensus and shared by the entire community.[18–22] There are several ways to represent such ontologies and many languages have been defined to represent them: from those based on first-order logic (FOL) (e.g., OWL,[23] RDF,[24] KIFF, etc.) to those frame-based languages implemented in ontology management systems (OMS) (e.g., Protege,[25] WebOde,[26,27] Ontolingua,[28] or WebOnto,[29] etc.).

Consequently, an ontology representing the structure of an *fuzzy DBMS* (FDBMS) facilitates the process of defining fuzzy information. This representation keeps the fuzzy metadata representation independent from the storage placement and allows the data to be represented from the special features of each DBMS. In

addition, by using an ontology, the fuzzy represented schemas is accessible via the semantic Web to all the users or agents that want to know its metadata.

There is wide discussion about the suitability of representing databases using ontologies, as we can see in Refs. 30–33. Many proposals for representing databases as ontologies have been developed, and examples can be found in Refs. 31, 34–36. Certain authors, however, consider database schemas to be lightweight ontologies[22] because they lack axioms which allow us to infer knowledge using them. Other authors consider both ontologies and databases to represent the same knowledge using different representation mechanisms.

Nowadays, there are many different schema representations available. In addition to relational database schemas, there are XML schemas, RDF schemas, or others from heterogeneous sources. All represent the structure of a certain information domain, and they can then be treated in a similar way, allowing information to be exchanged between them. Integration techniques enable this communication by solving the different schema conflicts that emerge when working with these heterogeneous sources. The exploitation of these integration techniques will allow information to be shared between both the existing and the fuzzy schemas (the intrinsic nature of which make the process quite flexible).

Our objective in this paper is to present an ontology that represents the fuzzy information of a fuzzy relational database. Owing to the formality of this representation, FMB access is more accessible to users or applications which use the ontology as an interface for access.

The paper is organized as follows: Section 2 of this paper briefly summarizes how fuzzy information can be represented both in general and in a database. This description also includes the problems found when this information is represented in a multifunction system. At the end of this section, a discussion about the suitability of using an ontology to represent databases is shown. Section 3 shows the proposed ontology to solve the problems described in previous sections, and its two subsections describe the two subontologies comprising the fuzzy data representation process. Section 4 describes how the ontology has been developed and the tools used to achieve this goal. Section 5 presents the advantages of data integration using this representation and the relation between this ontology and the semantic Web. Section 6 discusses the tool chosen for schema representation. Finally, Section 7 presents a summary of the advantages, limitations, and future lines of research for this proposal.

2. RELATED WORK

2.1. Fuzzy Data Bases Representation

Databases are the most extended knowledge representation system as a result of their efficient data management. Although relational databases were proposed by Codd[37,38] way back in the seventies, this is still the widest model used in the world for representing common databases[39] over others such as the object-oriented model. These databases do, however, present problems relating to the representation and handling of imprecise information.

The concept of imprecise information was formally defined by Zadeh in the fuzzy set theory,[40] which specifies that an element may or may not be a member of a crisp set or may have partial membership with regard to a set.[6] Imprecision, vagueness, and uncertainty are types of imperfect information in database and information systems. One kind of imprecise information is the null value,[5,41,42] which can be divided into

- *existing but unknown* value (denoted *unknown*),
- *nonexistent* value (denoted by *undefined*), and
- *no information* (*null*).

Existing fuzzy values are also described by fuzzy sets and possibility distributions.[40] The representation of fuzzy data in relational models is based on possibility distributions,[15,43] which appear as attribute values and also on relationships of proximity (or similarity or resemblance) which are associated with the data domains.[16,44,45]

Vagueness in a database system has been included in the system for two different tasks[5]: to admit vague queries to the classic databases[46,47] or to add vague information to the system.[15,16,43–45] The summary of DB systems has been extended to permit retrieval or representation of imprecise information.

Possibility distributions are not always represented by complex functions in a numerical domain. Possibility functions can be defined as more computationally efficient functions, e.g., triangular, interval or trapezoidal functions.[7] Linguistic labels can also be defined over any of these functions, avoiding any reference to the real representation of the data and bestowing them with a semantic sense. One of the most used possibility functions is the trapezoidal function, which is represented by means of four values: *alpha*, *beta*, *delta*, and *gamma*, where *alpha* represents the trapezoid lower left corner, *beta* represents the upper left corner, *delta* the upper right corner, and *gamma* the lower right corner, as shown in Figure 1. The possibility distribution shows the membership degree that any value of the universe has in the fuzzy set represented by this distribution. For example, we can define the possibility distribution that specifies a *teenager*'s age as [*11,13,16,18*], where the maximum membership degree (1) is reached in the *13–16* age range and the remaining ages

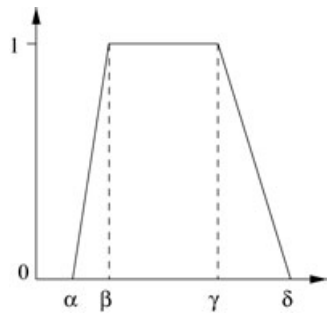

Figure 1. Trapezoidal possibility distribution.

Table I. Similarity relations for the behavior attribute.

	Indifferent	Calm	Loving	Restless	Eager	Aggressive
Indifferent	1	0.8	0.4	0.5	0.3	0
Calm		1	0.8	0.1	0	0
Loving			1	0.5	0.4	0.1
Restless				1	0.9	0.4
Eager					1	0.6
Aggressive						1

specified in the trapezoid, *11–13;16–18*, have a membership degree which increases or decreases, respectively, between [0,1]. The remaining values from this interval obtain a membership degree of 0.

On the other hand, not only do most relational database systems represent fuzzy information based on numerical domains with possibility distributions but they are also a representation describing discrete or scalar values in an ordered or unordered domain. A resemblance relationship can, however, be defined between the elements defined in the same domain. Two possibility distributions can then be represented in the same domain or discrete values in an ordered or unordered domain can be related by defining a resemblance relation between them. This relationship establishes the similarity degree that two elements in the domain have. Table I shows an example of how this relation can be defined.

Most of these extended proposals for representing vagueness and imprecision in database systems consist in adding an uncertainty degree to the tuple inserted in a table.[48,49] Other extensions allow a degree of certainty to be assigned to each value entered in the database, and on these, the database values may be represented as possibility distributions.[50,51] For example, we can store the following information: *Sally is a toddler* instead of saying *Sally is 2 years old*, where toddler is a label that represents an interval possibility distribution function with the value [*1*,*3*]. Another extension consists in representing the concept of similarity in a relational database.[52] This concept is based on the representation of perceptions (discrete values) that share a certain similarity between them, and so a person may be defined as *brown, light brown, fair*, or *dark*. Similarity relations must then be established between each discrete value, for example, we can fix that *brown* and *light brown* have a similarity degree of 0.8. On the other hand, an attribute of this domain can take a value such as "*someone is brown with a degree of 0.7*," and this sentence then relates to all the elements in the domain (*fair, dark*, etc.) with a certain degree.

Medina et al.[12] introduced the *GEFRED* model to represent flexible and imprecise information within the relational model. This model allows a relational attribute to store values such as nonnumerical discrete values, numerical values, a set of possible nonnumerical discrete values, a set of possible numerical values, possibility distributions on the basis of a nonnumerical domain, and possibility distributions on the basis of a numerical domain. This definition also allows three special values to be represented: *unknown* (when no concrete value is known), *undefined* (when the attribute is not applicable), and *null* (when the value can be unknown or undefined). The proposal of Medina et al. also includes a formal fuzzy language definition

Table II. Structures for representing fuzzy data types values.

Structure	Description	Example
Approx	represents an approximate value of the one given. The representation of this possibility distribution is equivalent to a triangle	*Annie is about five years old*
Interval	represents an interval value. The representation of the possibility distribution is like a square	*John is between 23–25 years old*
Trapezoidal	Represents a trapezoidal value. The representation of the possibility distribution is trapezoidal	*Adolescence is between 13–17 years old* (but the margin can be extended between 11–13 and 17–18 with a smaller degree)
Crisp	represents a common numerical value	*Mike is 1.72 meters tall*
Label	represents a value using a label. The value can be any of the remaining fuzzy data Type 2	*Mike is old*. Old is a trapezoidal value such as [65, 70, 80, 90]
Null	represents a null value	The value is unknown and undefined. It has no value
Unknown	represents an unknown value	The value is unknown
Undefined	represents an undefined value	The value is undefined

(FSQL) and an implementation of the model called FIRST. FSQL is based on the SQL standard, and the fuzzy data structures and operation management are represented using various new commands that have an equivalence in the SQL language.

The authors introduced the following three new data types into a classical RDMBS for representing this fuzzy information:

- *Fuzzy data type 1*, or *CRISP data type*, which allows attributes to store classic data which can be fuzzy queried;
- *Fuzzy data type 2*, or *POSSIBILISTIC data type*, which allows attributes to store fuzzy data using possibility distributions defined on a numerical domain; the information represented using this fuzzy data type can use the structures shown in Table II to store the data values.
- *Fuzzy data type 3*, or *SCALAR data type*, which allows attributes storing fuzzy data that are represented using possibility distributions defined on a nonnumerical domain. For example, we can define these values for a variable that represents a cat's behavior: *indifferent, Calm, Loving, Restless, Eager or Aggressive*. Relations between the different values are defined by a similarity degree, as shown in Table I.

To represent all these fuzzy data types, various tables must be included in the system catalog and these are called the *fuzzy metaknowledge base* (FMB).[53] These tables record all the fuzzy attributes, values, labels, and discrete values defined in the system, and the parameters implied by these definitions.

2.2. Extended System

Once fuzzy data types have been defined in the system, other extensions appear to exploit such data.[54] A proposal to deduce information stored in a fuzzy DBMS introduced various new relations in the FMB. These relations record new intensive relations, extensive relations, temporary tables, and other information relating to the deduction operation[55] and the fuzzy data types.

Meanwhile, another proposal for knowledge discovery using data-mining techniques appeared over fuzzy DBMS.[56] Fuzzy DBMS operations require new structures to be incorporated to the system catalog so that the information may be managed properly. As a result of all these extensions, the system catalog became impracticable and incomprehensible with the inclusion of so many difficult-to-understand structures in the catalog. New extensions to such a system require vast reconstruction of this catalog and a large number of resources.[54,57] The main problem, however, arises when the end user accesses this information since an alternative interface is necessary[58] which allows access to the DB information regardless of how this information is stored in the DBMS. This paper proposes an alternative that uses an ontology as the solution to achieve this task.[57,58]

2.3. Databases versus Ontologies

Ontologies are introduced in the semantic Web as the main mechanism for describing the content of a Web page.[59] This description can be made using different languages.[60–62] Most of these are based on first-order logic (FOL) (e.g., OWL,[23] RDF,[24] KIFF, etc.) and make the definition process very tedious. The most popular are the frame-based languages implemented in ontology management systems (OMS) such as Protege,[25] WebOde,[26,27] Ontolingua,[28] or WebOnto[29] among others. There are, however, drawbacks to all representation methods. FOL-based languages are too complex to be managed, but they are very good at inferring knowledge and they are independent of the application tool. OMS, meanwhile, represent a very easy ontology development interface, but representations are deeply dependent on the tool. These systems do, however, allow translations to be made into most FOL languages, thereby avoiding syntax mistakes. A detailed classification of ontology representation techniques can be seen in. Ref. 63.

Ontologies should provide consensual knowledge about a certain domain or area, and theoretically, these should be shared and populated so that this knowledge can be interchanged by the community. Such ontologies would allow common applications to be developed because of their compatible formats. Current work, however, demonstrates that each enterprise, project, or study develops its own ontology, uses its own language, and implements its own applications. General purpose ontologies (such as Cyc[64]) failed due to their low acceptance.[65,66] New trends in ontology representation are leading toward the integration of ontologies using matching and mapping processes.[67]

A large number of database-matching algorithms and studies have been revived to use similar developments with ontologies.[5,68–70] There is a great deal of debate about whether databases can be considered as ontologies when they represent a concrete domain knowledge. Some trends consider database schemas to be lightweight ontologies[22,30,71] because they lack the axioms which allow inferences to be made. Others consider ontologies and databases to be very similar, but they are not developed to represent the same aspect of the domain.[72] We consider database schemas to represent knowledge as ontologies do, but the resulting hierarchy could be a little flat and logical axioms could represent different kinds of restrictions (these are database

constraints). In general, however, they can be used to share stored information with the other semantic Web users and agents, and to profit from the new technologies developed around ontologies.

Many other proposals have been developed to enable database schemas to be accessed using ontologies. Most of these representations focus on populating the DB information in the semantic Web. These approaches only use relational schemas as a back-end system for retrieving data from databases through ontologies as a query interface.[33,73–75] The majority of these proposals define a declarative markup language for making the translation. Another uses a traditional closed program that establishes the mapping between the ontology and the database schema (e.g., Data Genie[76]). This last choice is obviously deeply dependent on the system and is nonscalable. The language-based method, however, is more independent of the system but programs also carry out the translation.

Another kind of proposal is that which attempts to represent database metadata (the schemas) as ontologies and this proposal is dealt with in this paper.

3. RESEARCH DESIGN

Many definitions of the *ontology* concept have been proposed in recent years.[22] Studer[20] defines an ontology as *a formal, explicit specification of a shared conceptualization*, where

- *formal* means that it is machine readable;
- *explicit specification* represents the concepts, properties, relations, functions, constraints, axioms that are explicitly defined;
- *shared* means that knowledge must be consensual; and
- *conceptualization* represents the fact that an ontology must be an abstract model and a simplified view of some phenomenon in the world that we want to represent;

and this definition summarizes the essence of an ontology. Other definitions for ontologies are very similar to the most referenced one which was given by Grubber[21] or Guarino.[18,19]

In the development process of an ontology that represents a fuzzy extension of the SQL standard, two ontologies must be defined. The first is defined to allow the schemas of a database to be represented as instances of this ontology. Classes of this ontology then represent the *fuzzy metaknowledge base* (FMB) of a database management system. Final data (the tuples of a relation) cannot, however, be defined using the instanced schema. Since instances can never again be instantiated, a new, second ontology, with classes representing the schema, must be defined where the values can be stored. In addition, all the fuzzy data structures that allow fuzzy information to be stored must be defined in this ontology as we will see in the following sections.

This proposal, however, goes further than merely representing SQL, but also establishes the bases for representing other operations that work with fuzzy data stored in databases such as data mining or fuzzy information deduction operations. This ontology has, therefore, been divided to establish the separation between the

concept of database schema and the management of the stored DB data. This division makes the following two *sub*ontologies more portable and reusable:

- *Database schema subontology*: This subontology describes all the schema elements specified in the SQL standard. These elements are only those which are related with the relational database representation. Some modifications to the standard are also included for representing fuzzy information as fuzzy columns and fuzzy data types. In brief, this subontology represents the metadata of the (fuzzy) extended SGBD catalog. The instances of this ontology define and manage DB structures such as tables, columns, constraints, index, etc. An example of a fuzzy SQL table definition, which can be made in this ontology, is

```
CREATE TABLE PLAYERS (
PLAYER VARCHAR2(60) NOT NULL,
TEAM   VARCHAR2(30) NOT NULL,
HEIGHT FTYPE2 NUMBER(3)... .
```

- *Fuzzy data subontology*: This subontology describes the schema of a relational database (the domain of a described problem) previously defined as instances in the *database schema subontology*. In its class hierarchy, it includes the classes that define the schema and the fuzzy data type structures defined in Section 2.1.[7] This last group of classes must be defined since this type of fuzzy data structure has never been defined before. Instances of this ontology enable users to define or manage data in the ontology. This ontology is to be used as an interface to make queries or definitions but never to store information; storage management is better accomplished in a DBMS. The following description shows the kind of information and the operations managed by this ontology:

```
INSERT into PLAYERS values ('P6' ,'Malaga',
'Very_tall',...)
```

  This example shows a tuple insertion in a fuzzy DBMS using the FSQL language.[53] An in-depth description of this subontology is beyond the scope of this paper.

3.1. Database Schema Subontology

An ontology that represents the SQL standard schema enables any database schema definition[77] to be independent of the DBMS where it is defined. In addition, once the schema has been translated into instances within the ontology, the schema can be imported or exported to another DBMS or could even be published on the Web so that its knowledge representation could be shared.

The SQL standard has been widely researched and Laborda[35] proposes the definition of only a few of the basic relational structures as a metaontology to communicate peer-to-peer databases. Trinh et al.[78] define most of the relational database structures as an ontology, and this representation includes the DB constraint definition as the semantic restriction of the ontology.

The proposal of Sujatha et al.[79] represents relational database schemas using a tool that translates them into OWL. All these proposals define their ontologies using OWL. Dou et al.[80] develop a process for representing the main relational database

structures in the semantic Web, and this proposal is based on its own declarative language. Calero et al.,[81] on the other hand, described the ANSI standard SQL 2003[77] using UML and defined it as an ontology to be used to represent relational database schemas. Another approach (Ontobase[82]) develops a tool that automatically represents database contents as a metaontology. Ontobase is a Protege plugin that imports database schemas to the Protege representation format.

In our proposal, we use the ontology of Calero et al.[81] to represent the standard ANSI SQL 2003, and in this ontology predefined data types and fuzzy data representation structures are defined. The formal definition of the fuzzy data structures allows the fuzzy data already stored in a DBMS to be represented.

The process of developing the *database schema ontology* is as follows:

- The ontology of Calero et al.[81] is pruned by eliminating all the object structures in the SQL 2003 schema definition.
- The resulting ontology is detailed with the predefined data types specified by the hierarchy of Pardede et al.[83] and an extension of fuzzy data types, as shown in Section 3.1.
- The merged ontology is modified to incorporate the fuzzy structures into the schemas. The resulting ontology description is detailed in Section 3.1

3.1.1. Data Type Definition

The SQL standard (in particular SQL:2003) defines three kinds of data types: predefined types (also known as "built-in data types"), constructed data types, and user-defined data types (further details of which can be found in Refs. 77, 81, 84). These data types have already been represented using different knowledge representation methods. One of these consists of the use of an ontology that models all the SQL:2003 data types using UML notation (see Ref. 81). However, this representation lacks explicit representation of the predefined data types and this can be found in Ref. 83, although fuzzy data type representation is not included.

The data type hierarchy presented in the work of Pardede[83] specifies all the predefined SQL data types as shown in Figure 2. The ontology presented by Calero et al.[81] represents all the SQL data types as an ontology and establishes the correspondence between them. The proposed ontology extends rather than modifies the SQL standard data types, as shown by the dashed line in Figure 2. The fuzzy data types represented in this correspond to those defined in the FIRST architecture mentioned in Section 2.1.

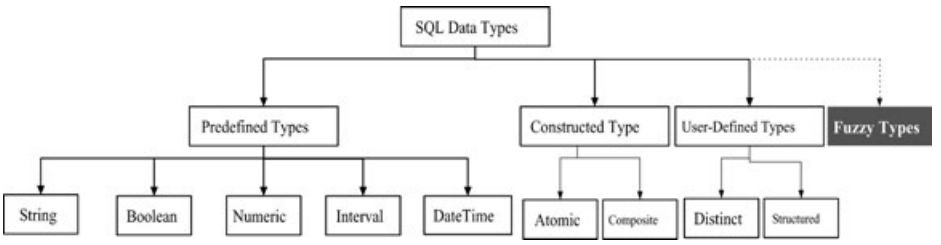

Figure 2. The taxonomy of Pardede[83] of predefined SQL data types and an extension with fuzzy data types with a dashed line.

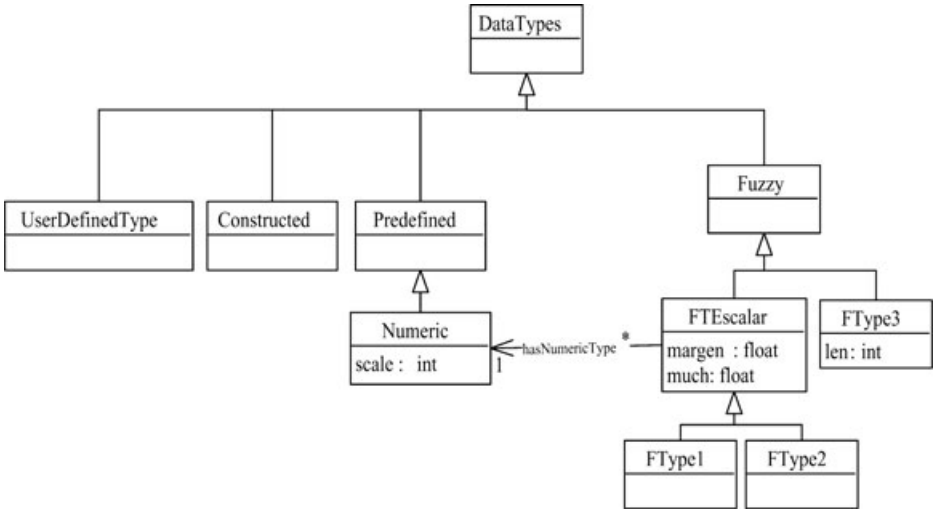

Figure 3. Fuzzy data type ontology.

Both representations are merged and the resulting ontology is then extended with the representation of the fuzzy data types. In Figure 3, the standard data types (those which are on the first level of the hierarchy) are presented with the fuzzy data types. The relation between the fuzzy data types and the classic data types is also detailed in Figure 3. More specifically, *fuzzy data type 2* is related with the *numeric* data type, which specifies that every fuzzy data type 2 has a numeric value structure associated with it.

3.1.2. Ontology Schema Description

Once the ontology of Calero et al.[81] has been pruned (object structures have been eliminated) and the data types have been defined, the ontology is extended to represent fuzzy information.

There is only one significant change and that is the *column class* is classified into two subclasses: *base_column* and *fuzzy_column*. The first represents classic columns and keeps all the same relations and attributes as the previous *column class* in the work of Calero et al. The second represents all the fuzzy columns represented in the DB. There is no relation between the *fuzzy_column* class and the other classes because the fuzzy attributes cannot be foreign keys or primary keys in the schema. In addition, it defines three new attributes called *NullableCharacteristic, UnknownableCharacteristic*, and *UndefinableCharacteristic*. All of these attributes represent whether the defined column can manage the *null, unknown or undefined* values, respectively. Figure 4 shows the resulting ontology.

In this ontology, the *Table* class is defined as a metaclass (it is a subclass of the *owl:class* in OWL). This metaclass also allows us to define all the tables described in the schema as classes.

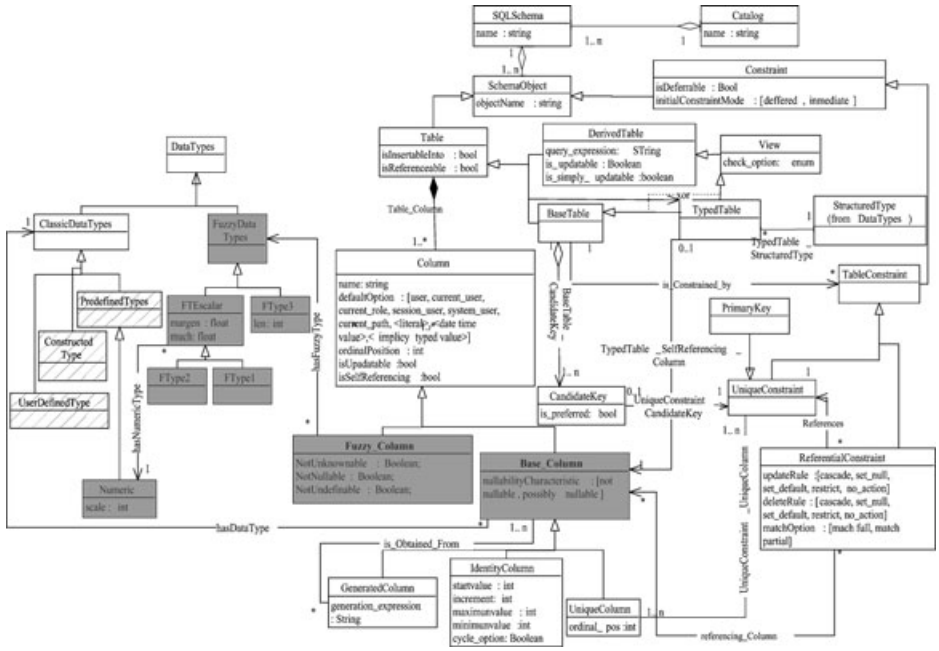

Figure 4. Schema ontology based on the work of Calero et al.[81]

This declaration is necessary to relate both subontologies (*schema and data subontologies*) since there is always correspondence between an instance in the *Table Class* in the *fuzzy schema ontology* and the corresponding class in the *data ontology* representing such a table.

The resulting ontology has been managed in two ways: first, through the use of a tool to represent ontologies (Protege[25]); and second, through the use of a language (OWL) and an API for managing it (JENA[85]). Both representations are described in the case study in Section 4.

3.2. Data Ontology

When the schemas are defined in the previous ontology as instances, the information representing this schema must be stored in a class representation. A new ontology must then be developed. The new ontology depends on the structures defined in the schema and some classes that represent fuzzy data structures enabling fuzzy values to be stored. There is no need to define classic data structures since they have already been defined and are known as XML data types. Correspondence between XML data types and SQL predefined data types is shown in Table III.

All the fuzzy data structures are based on predefined data types, but each representation has its own meaning.[8] A description of all these data structures and the remaining data ontology is shown in Figure 5. Each attribute defined in the ontology has a value (*hasValue* relation). There is a correspondence between each data type definition of the previous ontology with a fuzzy data structure defined in

Table III. Correspondence between SQL Data Types and XML Data Types.

SQL Data Type	XML Data Type		Facet
String, Bit, fixed	xsd:hexBinary xsd:base64Binary	or	no facet
String, Bit, varying	xsd:hexBinary xsd:base64Binary	or	xsd:maxLength
String, Character, Fixed	xsd:String		xsd:length
String, Character, Varying	xsd:String		xsd:maxLength
String, Character, clob	xsd:String		xsd:maxLength
Boolean	xsd:boolean		no facet
Interval	xsd:duration		xsd:pattern
Numeric, Exact, Numeric	xsd:decimal		xsd:precision, xsd:scale
Numeric, Exact, Decimal	xsd:decimal		xsd:precision, xsd:scale
Numeric, Integer	xsd:integer		xsd:maxInclusive, xsd:minInclusive
Numeric, SmallInt	xsd:integer		xsd:maxInclusive, xsd:minInclusive
Numeric, BigInt	xsd:integer		xsd:maxInclusive, xsd:minInclusive
Numeric, Approx, Real	xsd:float, xsd:double		no facet
Numeric, Approx, Double Precision	xsd:float, xsd:double		no facet
Numeric, Approx, Float	xsd:float, xsd:double		no facet
DateTime, Date	xsd:date		xsd:pattern
DateTime, Date	xsd:time		xsd:pattern
DateTime, T.Stamp	xsd:dateTime		xsd:pattern

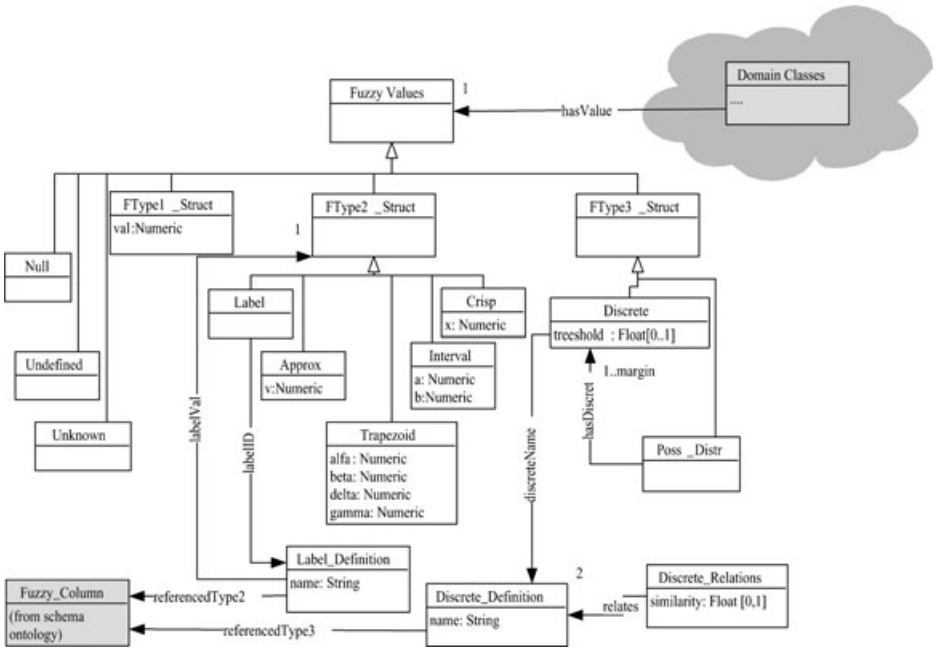

Figure 5. Data ontology. Fuzzy structure definition.

this ontology (Figure 5) or an XML data type shown in Table III. There are also other classes in the ontology that represent all the labels and discrete values in the domain, and these include the *label_definition* class, which allows all the labels to be controlled, and the *discrete_definition* class, which allows all the discrete values to be defined in the DB. The *discrete_relaton* class stores the relation between the different discrete values.

Rather than being developed to store information, this ontology has instead been created to construct queries or as an interface enabling better communication between users and the database. A DBMS is responsible for managing the information represented by the previous ontology.

4. CASE STUDY

The proposed ontology is defined by first developing the *schema subontology* (henceforth this will be called the *schema ontology*). The main reason for this is that metadata structures must be represented before definition of the schema structure enabling data to be stored.

There are two different definitions for the *schema ontology*:

- The first defines the ontology using an ontology-development tool and *Protege*[25,86] is the tool chosen for this purpose. This representation allows us to automatically generate the *data ontology* by developing a plugin for the tool. This tool provides the API which enables this utility to be developed and an example is shown in Section 4.1.
- The second defines the ontology using the ontology definition language OWL.[23,87] This choice focuses on publishing this ontology on the semantic web and making it accessible to most users (not only *Protege* ones). This representation is managed by using the API provided by JENA.[85] An example is shown in Section 4.2.

Once the ontology has been generated, a schema is defined by instancing the ontology. The example consists of a veterinary cat clinic which is defined using a UML class diagram as shown in Figure 6.

The resulting tables extracted from the UML representation are

```
CATS (  CatID       INTEGER PK,
        CatName     STRING (20),
        Age         FTYPE2 (1,2) FLOAT (1),
        Weigh       FTYPE1 (0.4,2.0) FLOAT (2),
        Character   FTYPE3 (3),
        hasBreed    (BREED.BreedName) )
BREED(  BreedName   STRING (100) PK,
        CharacterB  FTYPE3 (3))
VISIT(  Date        Date   PK,
        Price       FLOAT (2),
        Cat         (CATS.CatID) PK)
TREATMENT (
        illness     STRING (200)
        kind        FTYPE3 (2)
```

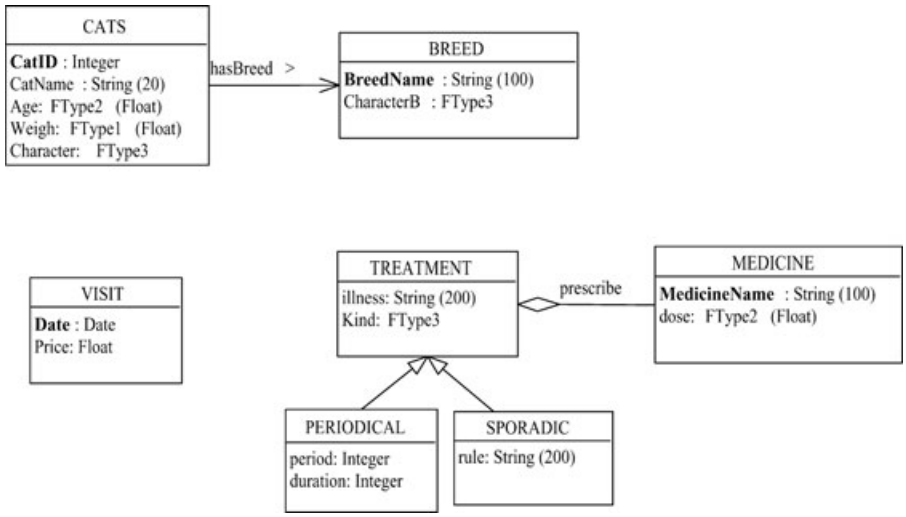

Figure 6. UML representation of the veterinary cat clinic example.

```
        Date          (VISIT.Date)   PK,
        Cat           (VISIT.CatID)  PK)
MEDICINE (
        MedicineName STRING (100) PK,
        dose FTYPE2   (0.5,3) FLOAT (2) )
PRESCRIBE (
        Medicine     (MEDICINE.MedicineName) PK,
        Date          (TREATMENT.Date)   PK,
        Cat           (TREATMENT.CatID)  PK)

PERIODICAL_TREATMENT (
        Date          (TREATMENT.Date)   PK,
        Cat           (TREATMENT.CatID)  PK,
        duration     INTEGER,
        period       INTEGER)

SPORADIC_TREATMENT (
        Date          (TREATMENT.Date)   PK,
        Cat           (TREATMENT.CatID)  PK,
        rule         STRING (200) )
```

4.1. Protege Representation

In this section, we will show how the schema ontology is defined using the *Protege* tool (Figure 7). Once the classes and relations of this ontology have been defined, the schema in the example is represented by instancing the corresponding classes. Some of these are shown in Table IV.

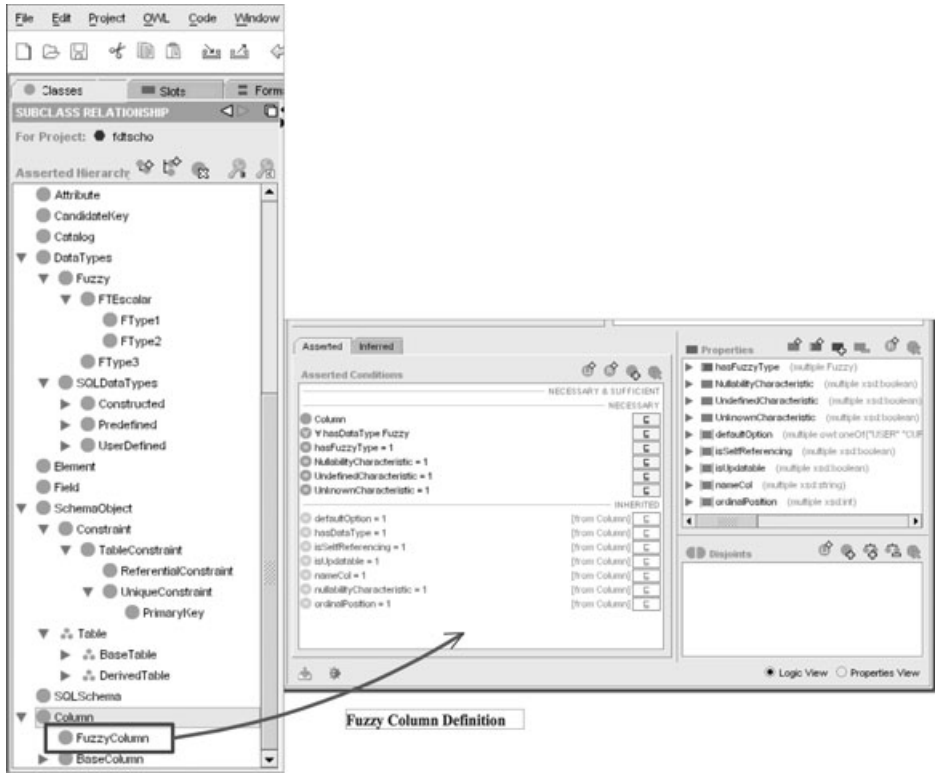

Figure 7. Protege schema ontology representation.

The plugin developed for the Protege tool attempts to automatically generate the data ontology corresponding to the previously defined schema. This implementation develops the following operations:

- Instances of the *Table Class* are also classes (since *Table* is considered a metaclass), but their properties must be defined. A new set of properties are generated for each column assigned to the ObjectProperty *hasColumns*.

Table IV. Some Instanced Classes for Defining the Cat Clinic Example.

Class	InstanceID	Description
BaseTable	Cats	Table Cats
CatID	UniqueColumn	Attribute of table Cats. This is Primary Key
PKCats	PrimaryKey	Name of Primary Key Constraint
CatAge	FuzzyColumn	Fuzzy Attribute of table Cats
FType2	FDT_CatAge	Fuzzy Data Type of Attribute CatAge
Character	FuzzyColumn	Fuzzy Attribute of table Cats
FType3	FDT_CatCharacter	Fuzzy Attribute of table Cats
hasBreed	BaseTable	Fuzzy Attribute of table Cats
ReferencialConstraint	FK_hasBreed	Foreign Key of attribute hasBreed
...	...	...

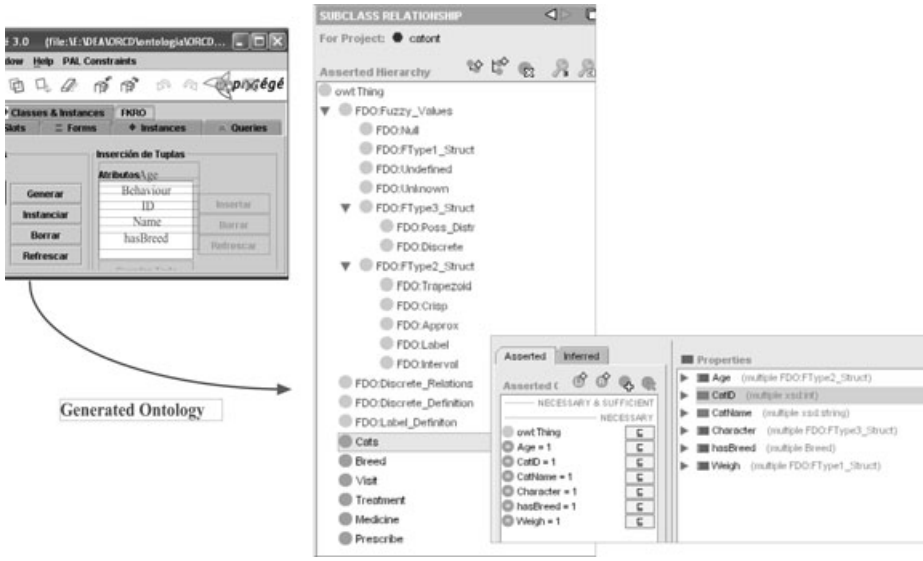

Figure 8. Plugin and schema ontology representation.

- Instances of the *Column* class are converted into ObjectProperties, the domain of which is the class representing the table where the column is. The range is the class corresponding to the data type of the column.
- Each datatype domain definition will establish the kind of XML value that each ObjectProperty has access to. If the DataType domain is fuzzy, then the range is the corresponding fuzzy structure defined in the data ontology.
- Cardinality restrictions limit the number of values that an ObjectProperty can have to one.
- Datatype properties of the FuzzyColumn class: NullabilityCharacteristic, UndefinabilityCharacteristic, and UnknowlabilityCharacteristic become a restriction for avoiding selecting the data type values null, undefined, or unknown, respectively.
- The range value of the foreign key attributes is the table class referenced by these attributes.

The new ontology is generated automatically using a plugin as shown in Figure 8.

4.2. OWL Representation

The schema ontology is also represented using OWL to make this ontology accessible for most users, not only those who work with *Protege*. In addition, an OWL representation allows the ontology to be published on the semantic Web and for users or agents to access it. Ontology users can then use the schema ontology to discover how fuzzy information is represented in fuzzy databases or they can access the schemas defined in this ontology.

Once the OWL schema ontology has been represented and the *Cats schema* has been defined on it, these ontologies are managed using the JENA[85] API which provides JAVA functions for managing the OWL ontologies.

```
<?x ml version="1.0"?>
<rdf :RDF
   xmlns :rdf ="http:// www.w3.org /1999/02/22- rdf -syntax- ns #"
   xmlns :xsd ="http:// www.w3.org /2001/ XMLSchema #"
   xmlns :rdfs ="http:// www.w3.org /2000/01/ rdf -schema#"
   xmlns :owl="http:// www.w3.org /2002/07/owl#"
   xmlns :fdtscho ="http:// personales.ya.com /fkrowl /fdtscho /fdtscho.owl #"
   xmlns ="http:// personales.ya.com /fkrowl /fdtscho /catschema.owl #"
   xml :base="http:// personales.ya.com /fkrowl /fdtscho /catschema.owl ">
<owl:Ontology rdf :about="">
 <owl:imports rdf :resource="http:// personales.ya.com /fkrowl /fdtscho /fdtscho.owl "/>
</owl:Ontology>

< fdtscho :BaseTable rdf :ID="Table_Cats">
  < fdtscho :hasColumns rdf :resource="# CatAge ">
  < fdtscho :hasColumns rdf :resource="# CatID ">
  < fdtscho :hasColumns rdf :resource="# CatName ">
  < fdtscho :hasColumns rdf :resource="# CatWeight ">
  < fdtscho :hasColumns rdf :resource="# CatCharacter ">
  < fdtscho :hasColumns rdf :resource="# hasBreed ">
</ fdtscho :BaseTable rdf :ID="Table_Cats">

< fdtscho :FuzzyColumn rdf :ID=" CatAge ">
  < fdtscho :nameCol rdf :datatype ="http:// www.w3.org /2001/ XMLSchema #string">Age
  </ fdtscho :nameCol >
  < fdtscho :hasFuzzyType >
    < fdtscho :FType2 rdf :ID=" FDT _CatAge ">
      < fdtscho :margin rdf :datatype ="http:// www.w3.org /2001/ XMLSchema #float">1.0
      </ fdtscho :margin>
      < fdtscho :much rdf :datatype ="http:// www.w3.org /2001/ XMLSchema #float" >2.0
      </ fdtscho :much>
      < fdtscho :hasNumericType rdf :resource="#DT_ CatAge "/>
    </ fdtscho :FType2 >
  </ fdtscho :hasFuzzyType >
</ fdtscho :FuzzyColumn >

< fdtscho :BaseColumn rdf :ID=" HasBreed ">
  < fdtscho :hasDataType rdf :resource="#DT_ BreedName "/>
  < fdtscho :nameCol rdf :datatype ="http:// www.w3.org /2001/ XMLSchema #string"
  > hasBreed </ fdtscho :nameCol >
 </ fdtscho :BaseColumn >
```

Figure 9. OWL code for Cat schema representation.

The developed tool implements two main operations:

- Translation of the defined schema in the ontology as instances in an SQL sentence. This sentence is capable of managing all the necessary catalog structures to define the fuzzy schemas in any RDBMS.
- A process that allows the DBMS catalog to be represented in any RDBMS that requires fuzzy data representation.

```
<fdts cho: ReferentialConstraint rdf:ID=" FK_hasBreed ">
  <fdtscho :referencedUnique rdf:resource="# PK_Breed"/>
  <fdtscho :ObjectName rdf:datatype ="http:// www.w3.org /2001/ XMLSchema #string"> FK_hasBreed
  </ fdtscho :ObjectName >
  <fdtscho :referencedColumn rdf:resource="# Col_HasBreed ">
  <fdtscho :constraintTable rdf:resource="#Table_Cats">
</fdtscho :ReferentialConstraint >
```

Figure 10. OWL code for Cat schema representation. Extension.

```
CREATE TABLE CATS(
 CatID INTEGER() PRIMARY KEY,
 CatName VARCHAR2 (20) NOT NULL,

 AgeT NUMBER(1) NOT NULL,
 Age1 FLOAT(1),
 Age2 FLOAT(1),
 Age3 FLOAT(1),
 Age4 FLOAT(1),
Weigh FLOAT(2),

 CharacterT NUMBER(1) NOT NULL,
 CharacterP1 NUMBER(3),
 Character1 NUMBER(3),
 CharacterP2 NUMBER(3),
 Character2 NUMBER(3),
 CharacterP3 NUMBER(3),
 Character3 NUMBER(3),

 hasBreed VARCHAR (100);
 CONSTRAINT FK_hasBreed FOREIGN KEY( hasBreed ) REFERENCES Breed( BreedName )
)
 // Fuzzy Attributes
 INSERT into Fuzzy_ Col_List values (Cats, AgeT ,2,1,");
 INSERT into Fuzzy_ Col_List values (Cats, Weigh,1,1,");
 INSERT into Fuzzy_ Col_List values (Cats, Character,3,3,");
//characteristic of attributes FTYPE1 and FTYPE2
 INSERT into Fuzzy_ Approx_Much values(Cats, AgeT ,1,2);
 INSERT into Fuzzy_ Approx_Much values(Cats,Weigh,0.4,2.0);
```

Figure 11. Returning SQL code for representing the Cats table in a DBMS.

A brief segment of the OWL code that represents the *Cat schema* example is shown in Figures 9 and 10. The developed tool uses a command prompt in order to execute its functions. A segment of SQL code returned from this tool that creates the *CATS* table in any DBMS is shown in Figure 11.

To represent a table in a Fuzzy DBMS, various operations need to be carried out. First, the table is defined with a variable number of columns; the number of columns depends on the parameters defined in each fuzzy attribute. Second, certain control data are stored in the fuzzy metaknowledge base (the extended catalog), FMB. These controls consist in knowing which columns are considered fuzzy and the parameters defined in each fuzzy data type instanced. *Fuzzy_Col_List* and *Fuzzy_Aprox_Much* are then two of these FMB relations.

5. MODEL USES

5.1. Integration of Information

Once the *schema ontology* has defined some fuzzy or classic database schemas as ontologies, the metadata and structure definition can be defined at any place (any RDBMS or similar) and published so that either intelligent agents or users can access these schemas and the information represented in them.

There are two basic operations when a database has been represented as an ontology: on one hand, the exportation process that allows an ontology to be generated from a database schema, and on the other, the importation process that allows a database schema to be generated using an ontology. These schemas from heterogeneous DB sources can be integrated using the proposed ontology as the communication layer.

By treating the schemas represented with this proposal as ontology data, users can then exploit them with all of the developed technologies for managing ontologies. Operations that can be run with ontologies include comparing, pruning, importing, exporting, evaluating, integrating, mapping, merging, alignment, and reasoning.

Merging is one of the most interesting and useful operations, which are possible with ontologies. The integration of schemas (sometimes called *fusion*) helps users unify information from many sources and use it in a joint way. Although information integration is a very complex problem, with this proposal some of these problems have been softened due to the fuzzy structures represented, with only those referring to information itself remaining. The following are examples of various integration problems with an explanation of their solutions:

Case 1: having a crisp relational DB and a crisp relational DB similar to the previous one (e.g., they might have different granularity in the representation of measures);
Case 2: having a crisp relational DB and a fuzzy relational DB representing the same reality;
Case 3: having two fuzzy relational DB representing similar realities.

In case 1, the problem can be solved by developing a new fuzzy database that establishes one kind of granularity and then adapts the included information as fuzzy data with a membership degree in keeping with the measure unit represented.

In case 2, the problem is resolved by adding the CRISP data to the fuzzy relational database with a membership degree of 1.

In case 3, information about the two fuzzy relational databases should be integrated by giving a different membership degree depending on the resemblance between the fuzzy column and the included data in it.

As we can see, representing fuzzy structures as an ontology can simplify the integration process of DB schemas.

5.2. Sematic Web Integration

The semantic Web[88,89] is ideal for sharing and exchanging information by means of semantic searches of the semantically described pages and resources

which are available online. Ontologies are one of the mechanisms for semantically describing web-accessible information.[90] Annotations are another mechanism which have also been proposed for this purpose as described by McCool in Refs. 66 and 91.

There are a wide variety of resources available on the Web (whether semantically described or not) including Web pages, different kinds of documents, applications, forms, lists, and even schemas that represent data structures. The most common schema representations are XML schemas, database schemas, and currently OWL[87] schemas or RDF.[24]

Various Web applications allow access to public databases and one example of this kind of application is the ISQLPlus,[92] a front-end application which accesses any database with published identification. Specification of a database's contents is, however, necessary so that these may be explored and this is a good reason for developing a tool to represent databases schemas and make them available on the web. OWL schemas or RDFS are the most widely used for describing database schemas on the semantic Web.

On the other hand, fuzzy data requires a particular representation of metadata to be published so that it may be accessed. The proposed ontology in this paper describes the metadata of such fuzzy information which allows users or applications to manage the fuzzy information stored in databases.[58] The ontology described in OWL allows the fuzzy data schemas to be published on the semantic web and the fuzzy data available in these fuzzy database systems to be accessed.

6. DISCUSSION

In this paper, the same ontology has been developed using two different technologies. On one hand, the *Protege* tool has been used to represent the *schema ontology* due to its intuitive interface for developing frame-based ontologies. This tool was chosen because it allows metaclasses to be managed, an uncommon feature in other tools such as *WebOde*, *OntoEdit*,[93] and many others. Protege also provides an API for developing other tools over the ontologies represented in this environment. The tool which is eventually developed will always depend on the Protege environment and the ontology has a specific representation which is only compatible with other *Protege* versions.

Representing an ontology using a knowledge representation (KR) language (namely OWL) does, nevertheless, make this representation independent and allows access to anyone wanting to access it. In addition, various APIs have been developed to enable ontologies to be managed in OWL, and *Sesame*[94] and *JENA*[85] are the most widely used.

OWL is not, however, without its disadvantages and this representation tends to make the output of these KR languages easy for machines to parse at the expense of human readability. Representing an ontology in OWL is, therefore, tedious because the language is complicated and the definitions are extremely long. Transcribing it manually is also impossible as mistakes are all too common. One solution might consist in using an OWL language editor or a development tool to generate the OWL

ontology, and then use it separately later. Most ontology-development tools include a translator to many ontology languages such as OWL, RDF, or KIF.

7. CONCLUSION AND FUTURE LINES OF RESEARCH

In this paper, we have proposed an ontology that defines a fuzzy relational database management system the (GEFRED model). This ontology represents the fuzzy database catalog that allows relational schemas to be defined to represent fuzzy and classic data. OWL is the language chosen to represent such an ontology and this allows the fuzzy DBMS structure and schemas to be included on the semantic Web. This inclusion adds new kinds of information to existing Web resources such as HTML documents, active pages, dynamic data, etc. Search agents can therefore obtain data from FDBMS automatically since the FDB structure specification is formally defined in a public ontology. The schemas that represent fuzzy data are also published on the web so that they can be accessed by the entire community.

Not all the semantic Web resources are semantically described as established by recent trends, and some of these Web resources are applications, which cannot be described semantically, or database schemas, which are described semantically using relational models. One example of this kind of application consists of front-end interfaces to databases, such as ISQLPlus©.[92] The existence of this kind of application justifies the publishing of fuzzy DB schemas.

The representation of a fuzzy data schema entails a hard learning process about how the system catalog is defined to define new fuzzy information on it. A fuzzy table representation has nothing in common with a common classic SQL table. The process is totally different since the number of relations implied in this process is larger. The problem becomes harder when new capabilities are added to the DBMS since the catalog has been extended. The schema ontology definition has simplified the process of defining fuzzy schemas because this ontology presents the concepts of this extension in addition to a specific DMBS representation. Moreover, this ontology presents the fuzzy extension concepts which are ordered, formally defined, and minimized.

By representing FDBMS as an ontology, not only can the fuzzy data catalog be easily exported to other DBMS but so can the defined schemas. The goal of the definition of the exportation process is to solve all the problems which exist when a migration is carried out between different DBMS representations (Oracle©, MySQL©, PossgreSQL, etc.). This is because each DMBS representation has its own special features (each DBMS representation does not have the same implementation of the SQL standard). On the other hand, the importation process means that a represented schema can change the DMBS representation. Using this ontology, it is possible to import and export schemas regardless of where the catalog has been constructed.

This representation is not, however, limited to describing standard SQL and classic database schemas and operations. This proposal establishes the basis on which other data models or operations can manage the already represented fuzzy information. Once the basic structures have been defined, new data types or operators

can, therefore, be easily described using the previous one and making more complex and functional ones. Some extensions to this basic proposal are described as a future line of this work.

Finally, fuzzy data representation can simplify the integration process of information from heterogeneous sources. By using ontologies as interfaces, different schema representations (e.g., XML schemas, DB Schemas or other schema from heterogeneous sources) can be integrated in the same way.

7.1. Limitations

An ontology is not, however, a data representation tool, and it is only defined to act as an interface between the user and the database due to the problems arising from this communication. Although the ontology attempts to organize the structures necessary for the formal representation of fuzzy information, it is not efficient when it comes to storing and managing the information stored in the represented database and so a close communication between the ontology and the database is, therefore, implemented in each developed function. Databases are in charge of managing the information.

On the other hand, the proposed ontology only represents information systems based on the relational model. The representation of information based on the object-oriented model is not supported by this ontology, although the standard SQL 2003 includes these capabilities in its definition.

Finally, this ontology lacks the representation of certain restrictions that are not allowed in the relational model. Deduction is not in fact the main objective of this ontology representation, and therefore various constraints about a real-world domain of the schema representation has not been considered in this proposal.

7.2. Future Directions

This ontology will be extended so that new capabilities may be added. Many proposals have been defined theoretically to enable fuzzy databases to make deductions using fuzzy data, and an additional proposal has been described and implemented to perform data-mining operations using this fuzzy data. Both extensions include new structures to be added in the data catalog of the fuzzy database representation of the theoretical model described in this paper. The ontology will provide a formal frame to represent a new and more complex catalog. This representation will act as an interface between the users and the database making these new capabilities more accessible and understandable.

A friendly interface that allows users to define their own fuzzy DB schemas, and data are another task to be developed. This interface should prevent users having to manage with OWL or specific ontology-development tools to define their own schema.

On the other hand, the *data ontology* enabling the schema to be represented as a set of classes provides an excellent interface for querying the database. An interface that helps the user to construct a query that will later be sent to the database is therefore a natural extension to this proposal.

The last goal proposed for this ontology is to develop a tool that solves the four types of conflicts for integrating schemas: i.e., naming conflicts, data type conflicts, data scaling conflicts, and missing data conflicts.[5,69] By solving these conflicts, fuzzy database schemas could communicate and exchange their data using the ontology as an interface. Other represented schemas can then be added to the integration process.

Acknowledgments

This research has been partially supported by the Andalusian Government (Spain) projects P06-TIC01433 and TIC-1570.

References

1. Zadeh LA. Knowledge representation in fuzzy logic. IEEE Trans Knowl Data Eng 1989;1(1):89–100.
2. Zadeh LA. Fuzzy sets, In Book: Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A. Zadeh; World Scientific Publishing Co., Inc. 1996. pp 19–34.
3. Hinde CJ. Fuzzy prolog. Intl J Man–Machine Stud 1986;11:569–595.
4. Lee J, Kuo J-Y, Xue N-L. A note on current approaches to extending fuzzy logic to object-oriented modeling. Int J Intell Syst 2001;16(7):807–820.
5. Ma Z. Fuzzy database modeling with XML. New York: Springer, 2005.
6. Ma Z. Fuzzy database modeling of imprecise and uncertain engineering information. New York: Springer; 2006.
7. Medina JM, Vila MA, Cubero JC, Pons O. Towards the implementation of a generalized fuzzy relational database model. Fuzzy Sets Syst 1995;75:273–289.
8. Blanco I. Deducción en Bases de Datos Relationales Difusas. PhD thesis, ETSI Informática, Universidad de Granada, Spain, 2001.
9. Klir GJ, Yuan B. Fuzzy sets and fuzzy logic, theory and applications. Upper Saddle River NJ: Prentice-Hall PTR; 1995.
10. Petry FE. Fuzzy databases: Principles and applications, international series in intelligent technologies. Norwell, MA: Kluwer; 1996.
11. Chen GQ. Fuzzy logic in data modeling; semantics, constraints and database design. Boston, MA: Kluwer; 1999.
12. Medina JM, Pons O, Vila MA. Gefred. a generalized model of fuzzy relational databases. Inf Sci 1994;76(1/2):87–109.
13. Galindo J, Urrutia A, Piattini M. Fuzzy databases modeling, design and implementation. Miami, FL: Idea Group Publishing; 2006.
14. Baldwin JF, Zhou SQ. A fuzzy relational inference language. Fuzzy Sets and Systems 1984;(14):155–174.
15. Raju KVSVN, Majumdar AK. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. ACM Trans Database Syst 1988;13(2):129–166.
16. Rundensteiner EA, Hawkes LW, Bandler W. On nearness measures in fuzzy relational data models. Intel Jl Approx Reason 1989;(3):267–298.
17. Chen GQ, Vandenbulcke J, Kerre EE. A general treatment of data redundancy in a fuzzy relational data model. J Am Soc Inf Sci 1992;(3):304–311.
18. Guarino N. Formal ontology, concept analysis and knowledge representation. Int J Human Comput Stud 1995;43(5/6):625–640.
19. Guarino N. Formal ontologies and information systems. In: Proc. of FOIS98; 1998. pp 3–15.
20. Studer R, Benjamins VR, Fensel D. Knowledge engineering: principles and methods. IEEE Trans Data Knowl Eng 1998;25(1/2):161–197.

21. Gruber TR. Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL 93-04, Knowledge Systems Laboratory, Standford University, 1993.
22. Gómez-Pérez A, Férnandez-López M, Corcho-García, O. Ontological engineering. New York: Springer-Verlag; 2003.
23. Antoniou G, van Harmelen F. Web ontology language: Owl.
24. W3C Recommendation. Resource Description Framework (RDF), February 1999. Available at http://www.w3.org/RDF/.
25. Stanford Medical Informatics at the Stanford University School of Medicine. Protege.
26. Ontological Engineering Group (OEG) from the Artificial Intelligence Department of the Computer Science Faculty (FI) from the Technical University of Madrid (Spain). Webode ontology engineering platform.
27. Arperez JC, Corcho O, Fernandez-López M, Gómez-Pérez A. Webode: a scalable workbench for ontological engineering. In K-CAP '01: Proceedings of the 1st Int Conf on Knowledge Capture. New York: ACM Press; 201. pp 6–13.
28. Stanford University. Ontolingua. http://www.ksl.stanford.edu/software/ontolingua/, April 2007.
29. Open University. Webonto. http://kmi.open.ac.uk/projects/webonto/, 2007.
30. Breu M, Ding Y. Modelling the world: databases and ontologies. Whitepaper by IFI, Institute of Computer Science, University of Innsbruck, 2004.
31. Meersman R. Ontologies and databases: More than a fleeting resemblance. Inf Technol Manag Ed. Springer, Rome Workshop, Luiss Publications; 2001;(6):97–122.
32. Spyns P, Meersman R, Jarrar M. Data modelling versus ontology engineering. In SIGMOD Record, 2002;12–17.
33. Dou D, LePendu P. Ontology-based integration for relational databases. In: SAC '06: Proc 2006 ACM Symp on Applied Computing. New York: ACM Press; 2006. pp 461–466.
34. Hakimpour F, Geppert A. Resolution of schematic heterogeneity in database schema integration using formal ontologies. Inf Techno Manage Ed. Springer, 2005;(6):97–122.
35. Pérez de Laborda C, Conrad S. Relational owl: a data and schema representation format based on owl. In CRPIT '43: Proc 2nd Asia-Pacific Conf on Conceptual Modelling. Darlinghurst, Australia: Australian Computer Society, Inc.; 2005. pp 89–96.
36. Partridge C. The role of ontology in integrating semantically heterogeneous databases. Technical Report 05/02, LADSEB-CNR; 2002.
37. Codd EF. The Relational model for database management, version 2. Reading, MA: Addison-Wesley; 1990.
38. Codd EF. Extending the database relational model to capture more meaning. ACM Trans Database Syst 1979;4:262–296.
39. La Disciplina de los Sistemas de Bases de Datos. José hernández orallo, 2002.
40. Zadeh LA. Fuzzy sets. Inf Control 1965;83:338–353.
41. Umano M, Fukami S, Mizumoto M, Tanaka K. Retrieval processing from fuzzy databases. Technical report, IECE, Japan, 1980.
42. Fukami S, Umano M, Muzimoto M, Tanaka H. Fuzzy databases retrieval and manipulation language. IEICE Tech Rep 1979; 78(233):65–72.
43. Prade H, Testemale C. Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries. Inf Sci 1984;(34):113–143.
44. Shenoi S, Melton A. Proximity relations in the fuzzy relational databases. Fuzzy sets Syst 1989;31(3):285–296.
45. Buckles BP, Petry FE. A fuzzy representation of data for relational databases. Fuzzy Sets Syst 1982;(7):213–226.
46. Kacprzyk J, Zadrozny S. Sqlf and fquery for access. IFSA World Congress and 20th NAFIPS Int Conf Joint 9th; 2001. Vol 4, pp 2464–2469.
47. José Tineo Rodríguez L. Extending rdbms for allowing fuzzy quantified queries. In: DEXA 2000. LNCS Vol. 1873. Berlin: Springer-Verlag; 2000. pp 407–416.
48. Baldwin JF. Fril-a fuzzy relational inference language. Fuzzy Sets and Systems 1987;14:155–174.

49. Zadeh LA. Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst 1978;1:3–28.
50. Prade H, Testemale C. Representation of soft constraints and fuzzy attribute values by means of possibility distributions in databases. In: Analysis of fuzzy information, JC Bezdek, ed., CRC Press; 1987. vol. 2, pp 213–229.
51. Umano M, Fukami S. Fuzzy relational algebra for possibility-distribution-fuzzy-relational model of fuzzy data. J Intell Inf Syst 1994;3:7–28.
52. Buckles BP, Petry FE. Fuzzy databases and their applications. In: Fuzzy information and decision processes, volume 2, chapter Amsterdam: North-Holland; 1982. pp 361–371.
53. Blanco I, Cubero JC, Pons O, Vila MA. An implementation for fuzzy relational databases. In: Bordogna G, Passi G, editors. Recent research issues on the management of fuzziness in databases; Studies in fuzziness and soft computing, Heidelberg, Germany: Physica-Verlag; 2000. pp. 183–207.
54. Blanco I, Martinez-Cruz C, Serrano JM, Vila MA. A first approach to multipurpose relational database server. Mathware Soft Comput 2005;12(2–3):129–153.
55. Blanco I, Martín-Bautista MJ, Pons O, Vila MA. A mechanism for deduction in a fuzzy relational database. In: Proc 9th Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002 Conference), Annecy, France; July 2002; pp 291–298.
56. Carrasco RA, Vila MA, Galindo J. FSQL: a flexible query language for data mining. Enterprise Information Systems IV, 2003. pp 68–74.
57. Blanco I, Martínez-Cruz C, Marín N, Vila MA. About the use of ontologies for fuzzy knowledge representation. In: Int Conf on Fuzzy Logic and Technology (EUSFLAT 2005); September 2005. pp 106–111.
58. Blanco I, Martínez-Cruz C, Vila MA. Looking for information in fuzzy relational databases accessible via the web, In: Handbook of research on web information systems quality, C Calaro, MA Moraga, M Piattini (Eds.), Idea Group Inc., 2008, Chapter XVIII.
59. Chandrasekaran B, Josephson JR, Benjamins, VR. What are ontologies, and why do we need them? IEEE Intellit Syst 1999;17:20–26.
60. Gómez-Pérez A, Férnandez-López M, Corcho-García, O. Metodologies, tools and languages for building ontologies. Where is their meeting point? Data Knowl Eng 2003;(46):41–64.
61. Duineveld AJ, Stoter R, Weiden MR, Kenepa B, Benjamins VR. Wondertools? a compartive study of ontological engineering tools. Int J Human–Cumpu Stud 2000; 52(6):1111–1133.
62. Su X, Ilebrekke L. A comparative study of ontology languages and tools. In: CAiSE 2002; 2002. pp 761–765.
63. WP8 Partners. Deliverable d8.1. State of the art and state of the practice including initial possible research orientations. Technical report, Network of Excellence, Contract no.: IST-508 011, 2004.
64. Lennat DB. Cyc: a large-scale investment in knowledge infrastructure. Commun ACM 1995;33(8):33–38.
65. Noy NF, Hafner CD. The state of the art in ontology design. A survey and comparative review. Am Assoc Artif Intell 1997; pp 53–74.
66. McCool R. Rethinking the semantic web, part I. IEEE Internet Comput, 2005;9(6):86–88.
67. Namyoun Choi, Song I-Y, Han H. A survey on ontology mapping. SIGMOD Rec., 2006;35(3):34–41.
68. Mena E, Illarramendi A. Ontology-based query processing for global information systems. Norwell, MA: Kluwer 2001.
69. Hai DH. Schema matching and mapping-based data integration. PhD thesis, Interdisciplinary Center for Bioinformatics and Department of Computer Science, University of Leipzig, Germany. 2005.
70. Staab S, Studer R. Handbook on ontologies. Berlin: Springer; 2004.
71. Noy NF. Semantic integration: A survey of ontology-based approaches. SIGMOD Rec 2004;33(4):65–70.
72. Unchold M, Gruninger M. Ontologies and semantics for seamless connectivity. SIGMOD Rec 2004;33(4):58–64.

73. Barrasa J, Corcho O, Perez AG. Fund finder: a case study of database to ontology mapping. In: Int Semantic Web Conference. Lecture Notes in Computer Science, vol 2870 Berlin: Springer-Verlag; 2003. pp 17–22.
74. Bizer C. D2r map—a database to rdf mapping language. In: 12th Int World Wide Web Conf, 2003, pp 17–22.
75. An Y, Borgida A, Mylopoulos J. Refining mappings from relational tables to ontologies. In Proceedings VLDBWorkshop on the SemanticWeb and Databases (SWDB04), August 2004. pp 57–62.
76. Gennari J, Silberfein, M. Datagenie. http://protege.cim3.net/cgibin/wiki.pl?DataGenie, 2007.
77. Information technology Ů database languages Ů sql. parts 1 to 4 and 9 to 14. 9075-1:2003 to 9075-14:2003 International Standards, September 2003.
78. Trinh Q, Barker K, Alhajj R. Rdb2ont: A tool for generating owl ontologies from relational database systems. In AICT/ICIW. Washington, DC: IEEE Computer Society; 2006. p 170.
79. Upadhyaya SR, Kumar PS. Eronto: a tool for extracting ontologies from extended e/r diagrams. In SAC '05: Proc 2005 ACM Symp Applied Computing. New York: ACM Press. 2005. pp 666–670.
80. Dou D, LePendu P, Kim S, Qi P. Integrating databases into the semantic web through an ontology-based framework. In: ICDEW '06: Proc 22nd Int Conf on Data Engineering Workshops (ICDEW'06). Washington, DC: IEEE Computer Society; 2006. p 54.
81. Calero C, Ruiz F, Baroni A, Brito e Abreu F, y Piattini M. An ontological approach to describe the SQL:2003 object-relational features. Comput Stand Interfaces J, 2005. pp 1–28.
82. Yabloko L. Next Generation Software. Ontobase plug-in for protege. http://www.ontospace.net/pages/3/index.htm, 2007.
83. Pardede E, Wenny Rahayu J. Impact of new sql standard to database modeling. Available at http://homepage.cs.latrobe.edu.au/ekpardede/paper/ENCYCLOPEDIA05-1.pdf.
84. Ansi/iso/iec international standard (is) database language sqlŮ part 2: Foundation (sql/foundation). ISO/IEC 9075-2:1999 (E), September 1999.
85. HP Labs Semantic Web Programme. Jena Ü a semantic web framework for java.
86. Knublauch H. An AI tool for the real world. Knowledge modeling with protègè. Technical report Available at http://www.javaworld.com/javaworld/jw-06-2003/jw-0620-protege.html.
87. Bechhofer S, van Harmelen F, Hendler J, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA. Owl web ontology language reference. Available at http://www.w3.org/TR/owl-ref/.
88. Berners-Lee T, Hendler J, Lassila O. The semantic web. Scientific American, 2001;284(5):28–37.
89. Goble C. The semantic web: an evolution for a revolution. Comput Netw 2003;42(5):551–556.
90. Sheth A, Ramakrishnan C, Thomas C. Semantics for the semantic web: The implicit, the formal and the powerful. J Semantic Web Inf Syst 2005;1(1):1–18.
91. McCool R, Rethinking the semantic web, part II. IEEE Internet Comput 2006;10(1):93–96.
92. Oracle. ISQLPLUS web enviroment. http://150.214.108.124/isqlplus, January 2007.
93. Ontoprise. Ontoedit datasheet 2003. http://electronicoffice.de/pdf/ontoprise/ontoedit data sheet.pdf, 2007.
94. Arjohn Kampman and Jeen Broekstra. Sesame. http://www.openrdf.org/, 2007.