A Framework for Fuzzy Ontology Storing onto Relational Database within an a Priori Data Integration System

El-Mehdi Berber
Computer Science Department
Oran University,
31000 Oran, Algeria

Beldjilali Bouziane
Computer Science Department
Oran University,
31000 Oran, Algeria

Myriam Lamolle
Computer Science Department
IUT de Montreuil, Paris 8 University
93100 Montreuil, France

ABSTRACT

Resolving semantic heterogeneity is still a challenging issue in data integration systems; but it can be strongly fixed when using ontology in an a priori approach where local ontology concepts are linked with shared ontology prior to populating data in corresponding sources. In this paper, we describe a defying context where local source is described by a fuzzy OWL ontology within an integration system using an a priori approach to achieve automatic integration for new data sources. We propose a conceptual framework starting by shared ontology and producing a target fuzzy Relational Database for every ontology-based local source participating in the integration system. Assuming shared ontology is a consensus in a given domain, this framework provides various contributions. It aims to solve ahead the problem of heterogeneous data sources because the local ontology that references the shared ontology is used to generate the conceptual data model for the target fuzzy Relational Database. To do this, it extends the a priori approach to deal with uncertainty which is a very common requirement in real world applications. Its storage process may be run on most of popular RDBMS. It is using a fuzzy OWL which represents most of fuzzy ontology constructs.

Keywords

data integration systems, database, description logic, ontology, fuzzy logic.

1. INTRODUCTION

Ontology is defined as a consensual and shareable conceptual model of a domain [1]. Unlike the conceptual model, which specifies the information must be represented in a database to answer a set of application loads; ontology is intended to describe all the information, in a consensual way, allowing fairly broad application areas to be conceptualized. Ontology provides essentially the knowledge of relevant terms of a domain. If a knowledge base of these terms was available and could be processed automatically, this should both accelerate the design process and make results more relevant. This should, also, reduce the amount of designer's work and would resulted in the development of more comprehensive and coherent conceptual models [2]. Furthermore, in the case of consensual domain ontology, it is possible to express different conceptual models corresponding to different specifications in terms of subsets or specialization of this ontology.

Several data integration approaches have concluded on the importance of ontology using to ensure semantic data integration automatically, especially to circumvent the issues of semantic heterogeneity, thus facilitating the specification of a conceptual model [3], [4]. Heterogeneity, which can characterize data sources used in data integration systems such as mediators or data warehouses, may be of different kinds: syntactic, systematic, schematic or structural. Actually, in classification of data integration systems with respect to automaticity criterion in [5], we conclude that automaticity can be strongly achieved when using ontology in an a priori approach then in a posteriori approach [6]. The latter means ontology is considered once the data sources have been set up. The former means that ontology concepts of local ontology are joined with shared ontology prior to populating data in corresponding sources. After local domain ontology is already defined from shared ontology, it can be used for target database design, and it will not be necessary to create another conceptual model for that design. This ensures that semantic heterogeneity is resolved upstream of integration process when new data sources must be added to the system.

To benefit from the automatic integration offered by ontologies, there are two main solutions. The first proposes to store whole ontology in addition to data on the target; the second proposes to transform all or part of the source ontology on the target. An example of the first solution is given in [7]. It does an explicit representation of local crisp ontology in each local database allowing their easier integration and even automatic generation of knowledge-level access interfaces. Such databases, combined with an ontology that defines the meaning, are called ontology-based databases (OBDB). An OBDB is composed of four parts; two of them are for storing classical data and two others for storing ontology data. This approach seems promising; but unfortunately, it adapts poorly to a fuzzy ontology model because of its ontology model that is inherent to the field of engineering domain and it can't be fuzzifed easily without changing its core syntax. Besides, with real ontologies having sizes increasingly large (hundreds of thousands concepts), it appears that storing the entire ontology will make a significant extra cost, especially this can greatly affect usability of the approach.

In the second solution, most of the proposed approaches [8], [9], [10], etc. propose to do direct linear transformation where associating a concept to a table and a property to a table attribute. This will induce, at least, as many tables in target RDB as there are concepts in source ontology, and will make a significant extra cost for transforming real ontologies having sizes increasingly large. Furthermore, the fuzzy extensions of ontology languages which have been presented are not complaint with OWL2 and current ontology editors [11].

To circumvent some of these drawbacks, we present, in this paper, a conceptual framework for fuzzy ontology storing on database. This framework assumes the existence of shared domain ontology as in an a priori approach for data integration system. This will guarantee automaticity since the domain ontology integrates different data sources feeding the future integration system to be developed. The proposed target schema is flexible and do not depend on source ontology size since it allows to add new resources with a cost of insertion or update in a classical DB. There is minimal impact of ontology update on target fuzzy DB size since this does not imply adding new tables. Also, storage process can be run on most of popular RDBMS and so database querying enjoys the power understood from the standard RDBMS query system. Actually, for a given application, a local ontology is first extracted from shared domain ontology, and then it is fuzzified and then transformed by a mapping process to produce the target fuzzy database. We have considered fuzzy ontologies in order to deal with uncertainty, which is a very common requirement in real world applications. But the conceptual formalism supported by crisp ontologies may not be sufficient to represent such uncertain information and many fuzzy extensions were proposed in literature [9]. We have chosen OWL2 syntax extended with fuzzy annotations properties as prescribed by [11], because it allows annotating, with fuzzy label annotations, most of OWL2 constructors, except for role constructors where it represents only fuzzy modified roles. A motivating example is provided later throughout the paper for well describing the proposed approach.

The rest of the paper is organized as follows. Section 2 presents a state of the art on the various approaches using ontology either to design conceptual models or to store fuzzy ontology in databases. Section 3 describes the process of fuzzy database design form fuzzy ontology and gives the overall structure of target RDB. Section 4 starts by giving an illustrative example which is then used to describe thoroughly the different steps in the process of storing a fuzzy Ontology on fuzzy RDB. Section 5 outlines formal proof of semantic preserving; Section 6 concludes the work and outlines some future perspectives.

2. RELATED WORK

2.1 Fuzzy Ontology

[12] proposes an extension of the description logic $\mathcal{SROIQ}(\mathcal{D})$ and provides a reasoning algorithm implemented to a prototype DELOREAN that supports fuzzy extensions of OWL and OWL2.

The authors of [13] define a fuzzy extension of the OWL language considering the possibility to add a concept modifier for a relation and introducing a new constructor to define belonging of objects to a given concept with a membership value greater or lower than a fixed value.

Among prominent and recent works on fuzzy ontologies, [11] extend most of OWL2 constructors with annotation properties representing features of the fuzzy ontology that OWL2 cannot directly encode.

2.2 Domain ontologies as conceptual models

In [14] the OWL-DL ontology is stored along with its instances in tables defined by the relational database system to prevent the loss of information. The transformation presented in [15] is based on a set of rules that specify how to map each construction in ontology to a corresponding structure in an Object-Relational database. In [16], ontology classes are mapped to relational tables, properties to relations and attributes, and constraints to metadata tables. Sugumaran, V. et al. in [17] show how a domain ontology, which captures knowledge on specific areas of application, can be used for the creation and validation of Entity-Relationship modeling for conceptual models. Architecture for an ontology management system is presented, and implemented in a prototype. [18] presents an ontology that can be used as a surrogate for the meaning of words in a database design system to simulate the contributions that a designer would make based on his/her general knowledge. The ontology classifies a term into one or more categories such as person, abstract good or tradable document. In [19], a domain ontology is described as an abstraction of the knowledge present in the data source schemas. A refining process is carried out for the ontology that is adapted to a local schema. The obtained ontology forms the starting point for the implementation of database schema.

2.3 Transformation of fuzzy ontologies to databases

Campaña, J.R. et al. in [10] define OWL ontology to allow fuzzy digital data types as the range of properties. They give algorithms used to convert the OWL ontology to a database schema. They also discussed the role of ontologies as tools for designing relational databases.

In [8], a set of storage rules, based on features of ontology structure and fuzzy ontology instances, is presented. Correctness of storage procedure for fuzzy ontology is evaluated, based on information capacity.

In [20] and [21], an ontology system is proposed to represent the knowledge structure enabling fuzzy information to be stored in fuzzy databases. Instances of ontology system represent diagrams describing the domain information in a database. These meta-models can be used to create the schema defined in different fuzzy DBMS according to the characteristics of available data representation.

[9] proposes an approach for storing items such as fuzzy classes, fuzzy properties and data instances of fuzzy ontology within fuzzy relational databases.

The authors in [22] propose a database schema to store ontology along with its instances preserving all information. Ontology and instances are stored in different schemas in order to improve the access to instances while retaining the capability of reasoning over the ontology.

In most of the approaches mentioned above, they propose to transform the complete ontology either into RDB tables or into an ER diagram which can be used then to create the target database. In most of these mentioned approaches, when the whole ontology must be transformed to a relational database, this will induce at least as many tables in target RDB as there are concepts in source ontology. If we take for example a local ontology with 25000 concepts, it will be very costly to manage 25000 tables in target database. Also, we must point out that there is little work that deals with this problem. Furthermore, to the best of our knowledge, there is no approach that proposes to automate the fuzzy RDB design in the context of a data integration system using an a priori approach.

In this paper, we describe the challenging context of a fuzzy OWL ontology as local source ontology within an integration system using an a priori approach in order to achieve automatic integration for new fuzzy data sources. We propose a conceptual framework starting by shared domain ontology and producing a target fuzzy RDB, for every local fuzzy ontology participating in the integration system.

The framework presented in this work assumes domain ontology is a consensus in a given domain and thus it provides various contributions. First, it aims to solve ahead the problem of heterogeneous data sources because the local ontology that references the shared ontology is used to generate the conceptual data model for the target fuzzy RDB. Secondly, this approach is using a fuzzy OWL which does not imply syntax change of basic OWL2 [11], and which represents most of fuzzy ontology constructs compared to related work. Thirdly, the proposed storage schema avoids highly large number of tables which could be implied by transforming each ontology concept and some object properties by a database table, as it is proposed by most of related work. Fourthly, it simplifies the designer's task and makes it more automatic and then much of time and efforts are saved. Fifthly, no special constraints are to be made on the target fuzzy database, since storage process has to be run on most of popular relational RDBMS and thereby database querying enjoys the power understood from the standard DBMS query system. Sixthly, transformation correctness is to be proven formally through information capacity model [23].

3. PROPOSAL OF A CONCEPTUAL FRAMEWORK FOR FUZZY DATABASE DESIGN FROM FUZZY ONTOLOGY

This section suggests how to use the proposed approach in the context of an a priori data integration system using fuzzy Ontology. Supposing a set of sources sharing domain ontology as described in [7], the framework proposed here suggests a set of steps (Fig. 1). First, we propose to extract source local ontology form shared domain ontology using, for example, approach proposed in [24]. After that, we use fuzzy OWL to add fuzzy annotations to each extracted source local ontology. At the end, the obtained fuzzy ontology is stored onto target fuzzy relational database. Hereafter are the steps:

1. Choice, selection and loading of a shared domain ontology O_p.
2. Data Base Administrator (DBA) extracts the class hierarchy for its own local ontology O_i; an approach is proposed in a previous work [24] for that extraction.
3. DBA links this class hierarchy C_i with that of the shared ontology C_p by defining the subsumption relationship between C_i and C_p.
4. DBA of each source chooses the classes, properties and individuals which will be fuzzified as required by the target application.
5. The DBA transforms the local crisp ontology to another fuzzy one using a dedicated ontology editor, or manually by using the fuzzy OWL syntax as described in the previous Section. This transformation is guided by a set of basic rules given in Fig. 3.
6. The local Fuzzy OWL ontology is stored into fuzzy Relational Database according to processes detailed in Section 4.2. The proposed storage target schema is divided into the following parts.

   a. **Ontology Structure Schema:** These are tables for describing all ontology resources with their namespace in source ontology (Table 1), classes/properties hierarchy (Table 2), properties restrictions (Table 3), properties characters (Table 4), and Fuzzy Resource types (Table 5).

   b. **Instance Mapping Schema:** Set of tables mapping ontology individuals with corresponding concepts and properties keeping membership degree (Table 6), with crisp and fuzzy properties (Table 7 and Table 8), with fuzzy data types (Table 9) and with fuzzy values (Table 10 and Table 11) in source ontology.

4. FUZZY ONTOLOGY STORING ONTO RDBMS

4.1 Illustrative example

For the sake of clarity, we shall give an example to illustrate the global steps of our proposed approach. First, a given classic domain ontology is chosen by a Database Administrator (DBA) in conjunction with a domain expert. Then, the DBA proceeds to extract local ontology which conforms to local application needs. After that, a fuzzification process is carried out to produce a fuzzy ontology (Fig. 2) according to predefined rules as seen above. Finally, the storing process is initiated to get the target fuzzy database as thoroughly explained in the subsequent sections.

4.2 Storing Ontology Structure

4.2.1 Storing Ontology Resource Table

It has structure depicted in (Table 1) and must describe all concepts, properties and individuals of the source fuzzy ontology. Onto_name column gives Ontology name; ID column identifies a resource, Type column differentiates resource types and URI column is intended to keep semantic articulation between RDB data and source ontology concepts. (Table 1) exhibits the result of storing process for classes (Fig. 4), properties (Fig. 5) and individuals (Fig. 6). The storing process was conducted on the ontology example given in Fig. 2, and its outcomes are shown progressively in the following paragraphs.

4.2.2 Storing Hirerachy Table

Table 2 keeps track of relationship between classes/subclasses and properties/sub properties. This table is filled by processes for storing ontology classes (Fig. 4) and properties (Fig. 5).

Table 2. Hierarchy Table

Res_ID1	**Res_ID2**	**Relationship**	**μ**
C3_AdminStaff	C2_Staff	SubClass of	1
C4_AcademicStaff	C2_Staff	SubClass of	1
C7_fzStaff_1	C4_AcademicStaff	SubClass of	1
C9_ma@tom.com	C8_Ma_email	SubClass of	0.2
C10_ma@yahoo.fr	C8_Ma_email	SubClass of	0.8

Table 1. Ontology Resource Table

Onto_Name	Resource_ID	Resource URI	Local Name	Type
Fuzzy_Local_Onto	C1_Department	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Department	Department	Class
	C2_Staff	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Staff	Staff	Class
	C3_AdminStaff	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74# AdminStaff	AdminStaff	Class
	C4_AcademicStaff	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74# AcademicStaff	AcademicStaff	Class
	C5_Student	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Student	Student	Class
	C6_Course	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Course	Course	Class
	C7_fzStaff_1	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#fzStaff_1	**fzStaff_1**	**Class = fuzzNominal Class**
	C8_Ma_email	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Ma_email	**Js_email**	**fuzzy Weighted Class**
	C9_ma@tom.com	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#ma@tom.com	**ma@tom.com**	**Class=base of C7**
	C10_ma@yahoo.fr	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#ma@yahoo.fr	**ma@yahoo.fr**	**Class=base of C7**
	OP1_Study_in	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Study_in	Study_in	ObjProperty
	OP2_Work_in	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Work_in	Work_in	ObjProperty
	OP3_Teach	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Teach_in	Teach	ObjProperty
	OP4_Choose_course	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Choose_course	Choose_course	ObjProperty
	fOP5_Email	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#email	**Email**	**fuzObjProperty**
	DP1_Staffname	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#staffname	Staffname	DataProperty
	DP2_Title	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#title	Title	DataProperty
	fDP3_Age	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#age	**Age**	**fuzDataProperty**
	Id_Staffid_1101	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74# Staffid_1101	**Staffid_1101**	**Fuzzyndiv=base of C7**
	Id_Depid_0206	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Depid_0206	Depid_0206	Indiv
	Id_Courid_309	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74#Courid_309	Courid_309	Indiv
	Id_fuzzMa_email	http://www.semanticweb.org/dell/ontologies/2014/11/untitled-ontology-74# fuzzMa_email	**fuzzMa_email**	**fuzzIndiv=base of C8**

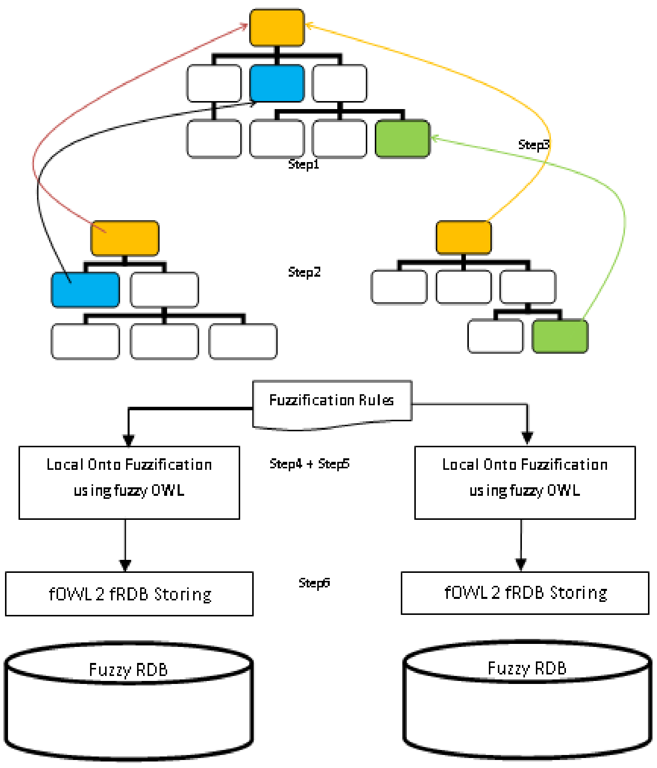

Fig. 1. Fuzzy DB design steps from fuzzy Domain Ontology

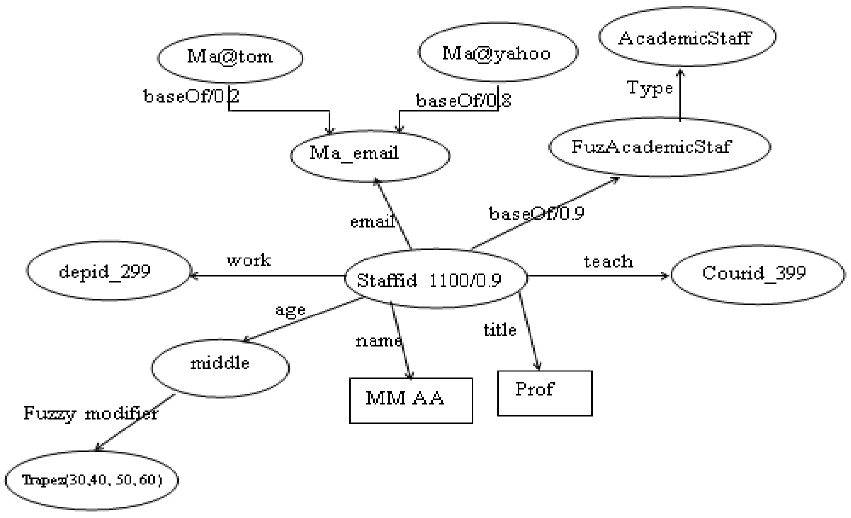

Fig. 2. Example of a Fuzzy Local ontology

1) For each fuzzy individual with membership degree <1
    a. Create a sub class named with fuzzy_RangeClassID+autoNumber.
    b. Add fuzzy Individual as base of fuzzy nominal concept of above created sub class with corresponding degree.
2) For each data Property Value for an individual having a possibility distribution as Range Type
    a. Create a root concept for each possibility item.
    b. Create a weighted sum concept having as bases the above created concepts with corresponding values.
    c. Make this individual as range value of concerned individual fuzzy value.
3) For each data property value with range equal to fuzzy label
    a. Create a fuzzy data type from the available fuzzy data types in fuzzy OWL.
    b. Assign created fuzzy data type to data property range.

Fig. 3. Fuzzification Rules

```
2) Find root Classes of source fuzzy ontology
3) Insert Classes into a queue
4) while Queue is not empty
       a. Pull queue head Class
       b. if Class is fuzzy then
              i. Add Class tuple in ‘fuzzy Resource Types Table’
       c. Add Class Tuple in ‘Resource table’
       d. If Class has sub-classes
              i. Insert Every Sub-class in queue
             ii. Insert a tuple in ‘Hierarchy Table’
5) End
```

Fig. 4. Storing ontology Classes

```
1) Find root properties of source fuzzy ontology
2) Insert properties into a queue
3) while Queue is not empty
       a. Pull queue head property
       b. if property is fuzzy then
              i. Add tuple for New fuzzy Property Range in ‘fuzzy Resource Types
                 Table’
       c. Add property in ‘Resource table’
       d. Insert tuple in ‘Property Restriction Table’
       e. Insert a tuple in ‘Property Specification Table’
       f. If Property has sub properties then
              i. Insert a tuple in ‘Hierarchy Table’ for each sub property
             ii. Insert every sub property in queue
4) End
```

Fig. 5. Storing ontology Properties

```
1) Locate individuals of source fuzzy ontology
2) Insert individuals into a queue
3) while Queue is not empty
       a. Pull queue head Individual
       b. Add individual in Resource table
       c. if individual is fuzzy then
              i. Add tuple in ‘fuzzy Indiv Per Class Table’ with fuzzy degree from its
                 super class Annotation in ‘fuzzy Resource Types table’
       d. Add fuzzy Property values in ‘FProp_Values_Per_Indiv Table’
       e. Add Tuple in ‘CrispProp_Values_Per_Indiv Table’
4) For each tuple in ‘FProp_Values_Per_Indiv Table’
5) If annotation assertion is Fuzzy Label then
              i. add tuple in ‘Fuzzy Label values Table’
6) If annotation assertion is possibility distribution then
              i. add tuple in ‘Possib_Distrib_values Table’ for each possibility
7) End
```

Fig. 6. Storing ontology Individuals

4.2.3 Storing Property Restrictions Table

Property Restrictions are stored in Restriction Table (Table 3) using process for property storing in Fig. 5. In addition to MaxCardinality and MinCardinality restrictions, one can store other property restrictions including: allValuesFrom, someValuesFrom, etc.

Table 3. Property Restrictions Table

Class_ID	Prop_ID	Type	Value
C4_AcademicStaff	OP3_Teach	mincard	1
C4_AcademicStaff	OP3_Teach	maxcard	3
C5_Student	OP4_Choose_course	mincard	2
C5_Student	OP4_Choose_course	maxcard	5

4.2.4 Storing Property Specification Table

Property characteristics are stored in the Property Specification Table (Table 4) according to property storing process in Fig. 5. Prop_Type column states if property is fuzzy/crisp data type, fuzzy/crisp object property. Character column states if property is transitive, functional, symmetric, etc. If ‘Property Range’ is a fuzzy one, then its range type is added, if not yet, in fuzzy types table (Table 5).

Table 4. Property Specification Table

Prop_ID	Prop_Type	Domain	Range	Character
OP1_Study_in	Obj	C5_Student	C1_Department	Functional
OP2_Work_in	Obj	C2_Staff	C1_Department	

OP3_Teach	Obj	C4_AcademicStaff	C6_Course	
OP4_Choose_course	Obj	C4_AcademicStaff	C6_Course	
fOP5_Email	FuzzObj	C7_fzStaff_1	C8_Ma_email	
DP1_Staffname	Data	C4_AcademicStaff	xsd:string	
DP2_Title	Data	C4_AcademicStaff	xsd:string	
fDP3_Age	FuzzData	C4_AcademicStaff	Fz_Age	

4.2.5 Storing Fuzzy Resource types Table

This table is filled according to processes for storing ontology concepts, properties and individuals. A resource is fuzzy if it has an annotation property fuzzyLabel as defined in [11]. A fuzzy Resource Type is inserted into this table in order to keep elements of its fuzzy Annotation Assertion. These elements which are filled here will be used to describe and fill fuzzy property values for each individual as done in (Table 9) which will be described hereafter.

Table 5. Fuzzy Resource Types Table

F_Type_ID	**F_Type_Onto_Name**	**F_OWL_Type_Name**	**F_OWL Annotation**
C7_fzStaff_1	fzStaff_1	Fuzzy Nominal	C7 Label
C8_Ma_email	Ma_email	Fuzzy weighted concept	C8 Label
fDP3_Age	Fz_Age	Fuzzy Data Type	Fz_Age Label

4.3 Storing Instance Mapping Schema

4.3.1 Storing fuzzy Individual Per Class Table

In (Table), μ denotes membership degree of a given individual to its class. If an individual is a crisp one, then its membership degree will be equal to 1. This table is filled by the process for individual storing (Fig. 6).

Table 6. Fuzzy_Individual_Per_Class Table

Indiv_ID	**Class_ID**	**μ**
Id_Depid_0206	C1_Department	1
Id_Courid_309	C6_Course	1
Id_fuzzJs_email	C8_ Ma _email	1
Id_Staffid_1101	C7_fzStaff_1	0.9

4.3.2 Storing CrispProp_Values_Per_Indiv Table

The Crisp property value of each individual is stored in this table by the process for storing individuals in Fig. 6.

Table 7. CrispProp_Values_Per_Indiv Table

Indiv_ID	**Prop_ID**	**CrispValue**
Id_Staffid_1101	OP2_Work_in	Id_Depid_0206
Id_Staffid_1101	OP3_Teach	Id_Courid_309
Id_Staffid_1101	DP1_Staffname	‘MMM AAA’
Id_Staffid_1101	DP2_Title	‘Prof’

4.3.3 Storing FProp_Values_Per_Indiv Table

For each value for a fuzzy object/data property, a tuple is created which contains a fuzzy value ID, individual ID, property ID. This task is done during individual storing process in Fig. 6.

Table 8. FProp_Values_Per_Indiv Table

Indiv_ID	**Prop_ID**	**fPropValueID**
Id_Staffid_1101	**fDP3_Age**	**fDP3_Age_ Id_Staffid_1101**
Id_Staffid_1101	**fOP5_Email**	**fOP5_Email_ Id_Staffid_1101**

4.3.4 Storing Map_fPropVal_fType Table

For each fuzzy individual property value, a tuple of its type identifier and its type literal is inserted in (Table 9) in order to map fuzzy type in (Table 5) with fuzzy values in (Table 10 and Table 11) seen below.

Table 9. Map_fPropVal_fType Table

fPropValueID	**fType_ID**	**fType_Literal**
fDP3_Age_ Id_Staffid_1101	ID_Fz_Age	middle
fOP5_Email_ Id_Staffid_1101	C8_Ma_email	fzMa_email

4.3.5 Storing fuzzy values of atomic type

A tuple of four classical parameters values for each fuzzy value having a fuzzy atomic data type as its range, are stored in this table. Comparison between fuzzy values can be done easily since they are all in the same table.

Table 10. Fuzzy_Label_Values Table

fPropValueID	**Alpha**	**Beta**	**Gamma**	**Delta**
fDP3_Age_ Id_Staffid_1101	30	40	50	60

4.3.6 Storing fuzzy values with Possibility distribution.

A tuple is created for every possibility in a fuzzy value which is of type fuzzy possibility distribution i.e. which is an instance of fuzzy weighted concept.

Table 11. Possib_Distrib_Values Table

fPropValueID	**fValue**	**fPossib**
fOP5_Email_ Id_Staffid_1101	Ma@yahoo.com	0.8
fOP5_Email_ Id_Staffid_1101	Ma@tom.com	0.2

5. PROOF OF SEMANTIC PRESERVING

The storing process described in Section 4 is a total transformation, as defined by [23], since source and target models are different and because it involves all source ontology components. This storing process is dominance preserving because every element in target fuzzy RDB has equivalence in source fuzzy ontology. Furthermore, schema translation is often combined with database integration or view integration. The framework proposed in this paper can be seen as a schema translation combined with DB integration as illustrated in Fig. 7.

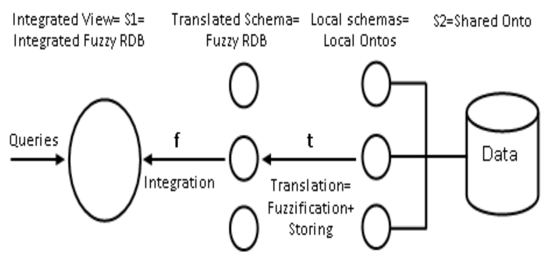
Integrated View= S1= Integrated Fuzzy RDB
Translated Schema= Fuzzy RDB
Local schemas= Local Ontos
S2=Shared Onto
Queries
f
Integration
t
Translation= Fuzzification+ Storing
Data

Fig. 7. Schema translation

Two schemas can be compared based on information capacity by using Schema Intension Graphs (SIG) formalism. The information capacity of a schema S is the set of all valid instances of S. Intuitively, a schema S2 has more information capacity than a schema S1 if every instance of S1 can be mapped to an instance of S2 without loss of information.

Target schema SIG. This schema in Fig. 8 is constructed as follows. While exploring ontology resource table (Table 1), a node is created for each class in ontology resource table. Then, by exploring hierarchy table (Table 2), a selection edge (labeled σ) is created for each concerned class tuple. For each class C, a projection node PN containing all C's data properties is created. A projection edge (labeled π) linking every property in PN with its data type range is added.

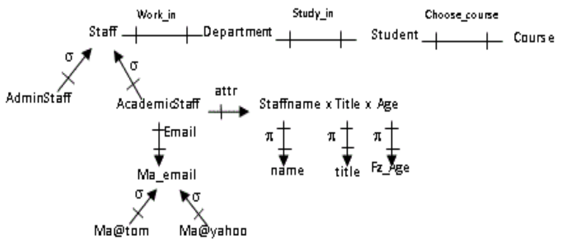

Fig. 8. Target SIG

Source schema SIG. Fig. 9 gives the SIG schema with respect to source local ontology described in Fig. 2. A selection edge represents subsumption relation between two classes. Under constraint of disjointness and completeness for *AdminStaff* and *AcademicStaff*, a sum node *AdminStaff + AcademicStaff* is created and linked by a bijective selection edge with Staff node. Edge *work_in* is functional since each staff works in a single department. Projection edges link projection node attributes with their data values.

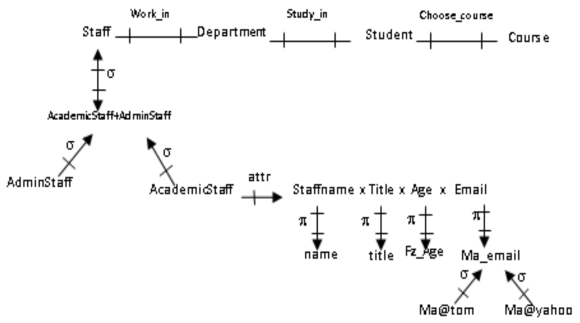

Fig. 9. Source SIG

One way to align the two SIGs is to clear the edge *email* in Fig. 8, and add extra attribute in the product node with a projection edge towards *Ma_email* node. Also, a sum node for *AdminStaff* and *AcademicStaff* can be added to target schema for the same reason it was created in source schema. This would make equivalent source and target SIG schemas.

6. CONCLUSION

Until now, there is no standard for crisp/fuzzy ontology storing onto RDB, although ontologies are more and more present in all knowledge areas. Moreover, there are still only few approaches that deal of storing fuzzy ontologies on databases albeit fuzzy ontologies play an increasing important role for dealing with uncertainty, which is a very common requirement in real world applications.

To tackle this issue, we presented, in this paper, a conceptual framework for fuzzy database design from shared domain ontology including fuzzy ontology storing onto RDB. This design approach assumes the existence of a shared ontology, in an a priori approach, integrating different data sources feeding the future integration system to be developed.

This approach gets several advantages that need to be mentioned. First, the transformation of the whole ontology is not required, thereby saving time and space. Secondly, chosen fuzzy extension for source ontology language OWL does not imply syntax change of basic OWL2 [11], and it represents most of fuzzy ontology constructs compared to related work. Thirdly, target fuzzy database has to be run on most of popular relational RDBMS and so database querying enjoys the power understood from the standard DBMS query system. Furthermore, over other approaches, our approach provides independence of the target database size, which does not depend on considered domain ontology. Nevertheless, this work could be improved by considering the following aspects:

─ Study of the impact of changes in fuzzy sources ontology on the target fuzzy database;

─ Thorough study of the reasoning problem in source ontology and target database. Reasoning on OWL ontology is still a thorny track because of N2ExpTime [25]. For example, a full reasoning algorithm for [11]'s fuzzy logic is not known yet. They propose a parser to translate fuzzy ontology to a language adapted to fuzzy DL reasoners such as fuzzyDL or DeLorean. While in [10], they propose that some basic reasoning abilities can be added to the system through the combined use of the original ontology and the instance data in the database schema.

7. REFERENCES

[1] Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge acquisition 5, 199-220 (1993)

[2] Tho, Q.T., Hui, S.C., Fong, A.C.M., Cao, T.H.: Automatic fuzzy ontology generation for semantic web. Knowledge and Data Engineering, IEEE Transactions on 18, 842-856 (2006)

[3] Noy, N.F.: Semantic integration: a survey of ontology-based approaches. ACM Sigmod Record 33, 65-70 (2004)

[4] Bellatreche, L., Dung, N.X., Pierra, G., Hondjack, D.: Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. Computers in Industry 57, 711-724 (2006)

[5] Bellatreche, L., Pierra, G., Xuan, D.N., Hondjack, D., Ameur, Y.A.: An a priori approach for automatic integration of heterogeneous and autonomous databases.

In: Database and Expert Systems Applications, pp. 475-485. Springer, (2004)

[6] Pierra, G.: The PLIB ontology-based approach to data integration. Building the Information Society, pp. 13-18. Springer (2004)

[7] Dehainsala, H., Pierra, G., Bellatreche, L.: Ontodb: An ontology-based database for data intensive applications. Advances in Databases: Concepts, Systems and Applications, pp. 497-508. Springer (2007)

[8] LV, Y., ZHANG, D.: Semantic Preserving Storage Approach of Fuzzy Ontology. Journal of Computational Information Systems 8, 8675-8682 (2012)

[9] Zhang, F., Ma, Z., Yan, L., Cheng, J.: Storing fuzzy ontology in fuzzy relational database. In: Database and Expert Systems Applications, pp. 447-455. Springer, (2011)

[10] Campaña, J.R., Medina, J.M., Vila, M.A.: Semantic Data Management Using Fuzzy Relational Databases. Flexible Approaches in Data, Information and Knowledge Management, pp. 115-140. Springer (2014)

[11] Bobillo, F., Straccia, U.: Fuzzy ontology representation using OWL 2. International Journal of Approximate Reasoning 52, 1073-1094 (2011)

[12] Bobillo, F., Delgado, M., Gómez-Romero, J.: Crisp representations and reasoning for fuzzy ontologies. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 17, 501-530 (2009)

[13] Calegari, S., Ciucci, D.: Fuzzy ontology, fuzzy description logics and fuzzy-OWL. Applications of Fuzzy Sets Theory, pp. 118-126. Springer (2007)

[14] Das, S., Chong, E.I., Eadon, G., Srinivasan, J.: Supporting ontology-based semantic matching in RDBMS. In: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, pp. 1054-1065. VLDB Endowment, (2004)

[15] Astrova, I., Korda, N., Kalja, A.: Storing OWL ontologies in SQL relational databases. International Journal of Electrical, Computer and Systems Engineering 1, 242-247 (2007)

[16] Vysniauskas, E., Nemuraite, L.: Transforming ontology representation from OWL to relational database. Information Technology and Control 35, 333-343 (2006)

[17] Sugumaran, V., Storey, V.C.: The role of domain ontologies in database design: An ontology management and conceptual modeling environment. ACM Transactions on Database Systems (TODS) 31, 1064-1094 (2006)

[18] Storey, V.C., Dey, D., Ullrich, H., Sundaresan, S.: An ontology-based expert system for database design. Data & Knowledge Engineering 28, 31-46 (1998)

[19] Roldan-Garcia, M., Navas-Delgado, I., Aldana-Montes, J.F.: A design methodology for semantic Web database-based systems. In: Information Technology and Applications, 2005. ICITA 2005. Third International Conference on, pp. 233-237. IEEE, (2005)

[20] Blanco, I.J., Vila, M.A., Martinez-Cruz, C.: The use of ontologies for representing database schemas of fuzzy information. International Journal of Intelligent Systems 23, 419-445 (2008)

[21] Martínez-Cruz, C., Blanco, I., Vila, M.A.: Describing Fuzzy DB Schemas as Ontologies: A System Architecture View. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications, vol. 81, pp. 147-157. Springer Berlin Heidelberg (2010)

[22] Barranco, C.D., Campaña, J.R., Medina, J.M., Pons, O.: On storing ontologies including fuzzy Datatypes in relational databases. In: Fuzzy Systems Conference, pp. 1-6. IEEE, (2007)

[23] Miller, R.J., Ioannidis, Y.E., Ramakrishnan, R.: The use of information capacity in schema integration and translation. In: VLDB, pp. 120-133. Citeseer, (1993)

[24] Berber, E.-M., Beldjilali, B., Myriam, L.: Towards an ontological framework for the automatic generation of relational database schemas. In: Colloque sur l'Optimisation et Systèmes d'informations (COSI'2014). (2014)

[25] Kazakov, Y.: RIQ and SROIQ are harder than SHOIQ. In: In Proc. KR’08. (2008)