Towards a Model for Multimedia Dataspaces

Armelle NDJAFA, Harald KOSCH, David COQUIL
Chair of Distributed Information Systems
University of Passau
94032 Passau, Germany
{firstname.lastname}@uni-passau.de

Lionel BRUNIE
Lyon University, CNRS
INSA-Lyon, LIRIS, UMR5205
F-69621, France
{firstname.lastname}@insa-lyon.fr

Abstract —many domains are facing challenging issues due to the volume, complexity and diversity of the digital information that they hold. Classical data integration techniques can hardly be applied in this context to combine information residing at different sources and to provide the user with a unified view of the information as they cannot cope with the complexity and the volume of the data. Among the proposals that were made in this domain, the dataspace paradigm is currently taking the first place in research. In this paper, we introduce a dataspace representation model. The model defines the dataspace as a set of classes, objects and relations. It addresses a number of shortcomings of existing models. In particular, it covers a large set of relation types, enabling the integration of many types of information such as structured and semi-structured data sources and ontologies. It also addresses the needs of multimedia data integration by providing means to represent similarity-based relations between multimedia objects.

Multimedia Dataspace; Data Management; Data integration; Modelization

I. INTRODUCTION

Many domains are currently facing an explosion of the amount of digital information and of data sources [1]. Progress in network technology enables the interconnection of the growing number of distributed data sources. Modern systems in fields like medicine, Web publication management or personal data management must be able to simultaneously support large numbers of distributed data sources characterized by a high level of data heterogeneity, for example in terms of file formats (multimedia documents, PDF, XML, HL7, etc.), storage model and degree of structure (Multimedia databases, relational databases, XML databases, etc.). A difficult challenge for such systems is to provide their users with a global view of the data, which the users can use to formulate queries. Classical data integration solutions proposed for similar issues are hardly applicable in this context. Indeed they were designed for complete structured data and typically rely on the computation of a global schema. They are not adapted to the scale of the problem. Moreover, the existing technology can neither deal with rapidly changing resources nor implement on-the-fly data integration [2]. In view of these shortcomings, the concept of dataspaces has been proposed in [3], [4] as a new abstraction for large-scale heterogeneous and distributed data management. Dataspaces do not require up-front efforts to semantically integrate the sources before base services (e.g. keyword searches) on the data can be provided. They support uncertainty in schema mappings, taking into account the fact that a mediated schema may not be available, and that the schema mappings between the sources and the mediated schema may be inaccurate. Another interesting feature of a dataspace is gradual data integration. The system incrementally integrates the data over time depending on user needs (i.e. based on their queries), possibly with the assistance of the users to confirm matches as the system runs.

Dataspaces research is currently very active. Many works target the improvement of specific dataspace components [5], or define complete dataspace management systems [6]. Currently, the most active research topics in the domain are pay-as-you-go user feedback [5], dataspace schema matching and mapping [7] and query evaluation [8]. Despite this activity, existing models have a number of shortcomings that limit their applicability. First, they tend to overlook the different types of relations that can exist between data items, restricting the amount of information that they can practically integrate. Moreover, they target classical textual data sets without considering the specificities of multimedia data. Finally, they do not provide the fine-granularity in the persistence of integrated data that is needed in specific domains.

To address these issues, we are developing a dataspace model, which defines a dataspace as a set of classes and relations between these classes. The data can originate from different types of data sources, knowledge representation resources such as ontologies, or canonical knowledge. The envisioned application domain being e-health, we present the model using medical domain examples throughout the paper; however the work is generic and can be applied to other domains. One of the design goals of the model is to maximize its expressivity in terms of the types of relations that it can represent, enabling it to deal with information originating from structured data, semi-structured data, ontologies and other forms of knowledge representation as well as from canonical knowledge. In particular, we include *similarity relations* in the type of relations that we consider. Similarity relations are a specificity of multimedia data representing non-exact matching between objects. In term, identified similarity relations can be used to derive relations between other objects in the dataspace. For example, two patients with closely matching mammographic scans may have the same type of disease. In addition, the model introduces the concept of *dataspace views*, which enables to store the results of queries on a dataspace as a new dataspace with different modes of persistence (virtualized view, materialized view, mode of synchronization with the content of the original sources…).

The model was not originally conceived for the Web. However, it is generally consistent with the vision of the "Web

of data". Indeed, in [9], the authors argue that the Web of Data corresponds in essence to the creation of a dataspace using Web technology. More precisely, our model follows previous dataspace work by Halevy et al. [3] in representing a dataspace as a set of triples of the form (object, attribute, value). We are thus planning to implement it using RDF graphs and SPARQL as a query language. This will make the model consistent with the Linked Data paradigm, and thus improve its applicability.

The remainder of this paper is organized as follows. In Section II, we define the dataspace model by giving the formal definition of classes and relations. Section III outlines the representation of similarity relations in our model. Finally, in Section IV, we draw conclusions and present future work.

II. DATASPACE MODEL

In this section, we formally define a dataspace as a set of classes, objects, and relations between them. Classes and relations are derived from information contained in data sources, ontologies or canonical knowledge. Canonical knowledge enables knowledge to be manually added to the dataspace even if it is not directly defined in any source.

A. *Classes*

A class C is defined as a 4-tuple (n_C, cs, d_C, A_C), such as:

- n_C is the name of the class.
- *cs* is the class-set to which the class C belongs.
- d_C is the date of the class synchronization (insert- or update-date) with the original source.
- A_C is an optional set of attributes {(attr, type)}, where each attributes has a type.

In our framework, classes are defined in the sense of classes in ontologies. Classes have different origins; they may come from unstructured data sources, data sources schemas, ontologies or from canonical knowledge. For instance, a table in a relational database is represented as a class, a class in the ontology is also represented as a class and information from canonical knowledge may be represented as a class. Classes originating from the same source are grouped into a class-set.

The class-set denoted cs is defined as a 3-tuple (n_{cs}, o_{cs}, loc_{cs}), such as:

- n_{cs} is the name of the class-set.
- o_{cs} is the origin type of the source, which can take three values: Data source, Ontology or Canonical knowledge
- loc_{cs} is an URI identifying the data source

A class comprises two types of attributes: class attributes and instance attributes. n_C, *cs*, d_C are class attributes taking the same values for all instances of the class. Instance attributes (A_C) describe an instance and may take different value for each instance. Each attribute has a unique type, which is a primitive data type. For instance, the class Patient has the attribute (age, integer), (name, string), (birthdate, date), (disease, string).

B. *Objects*

An object is an instance of a class. The attributes can have values of the type specified in the class definition. For instance, referring to a class Patient, an object of this class can be $patient_1$ with attributes {(age, 45), (name, Paula), (birthdate, 15/05/1967), (disease, Breast Cancer)}.

C. *Relations*

1) Definition

A relation is defined as a 5-tuple (n_R, e_1, e_2, p_R, A_R) such as:

- n_R is the name of the relation
- e_1, e_2 are entities that are put in relation. An entity is either a class or an object
- p_R groups the properties of the relation
- A_R is an optional set of attributes {attr}

p_R, the property of a relation is a 5-tuple (t_R, m_R, c_R, loc_R, cond), such as:

- t_R is the relation type: CRC, ORC or ORO
- m_R is the relation modus: IRelation or ERelation
- c_R is the relation category: Classification, Meronymic, Similarity, Temporal, Spatial or Any
- A_c is an optional set of attributes
- loc_R identifies the source of the relation as an URI
- *cond* is the optional collection of validity condition which may be:
  - a set of temporal conditions {t}
  - and/or an uncertainty condition (u)

We define three types (t_R) of relation: class to class (CRC), object to class (ORC) and object to object (ORO) relation. The relation modus (m_R) of the relation can be defined as internal and/or external. Internal relations (IRelation) are relations between classes or between objects derived from the same data source. They are mostly relations existing in the data sources like foreign keys in databases. External relations (ERelation) are on the contrary defined between classes, objects belonging to different class-sets. These relations are gradually discovered during the operation of the dataspace.

Based on this definition, CRC relations can be internal or external. ORC relations on the other hand are always internal. ORC relations correspond to instance_of relations (e.g. $person_{Paula}$ *instance_of* Person or Person *has_instance* $person_{Paula}$). ORO relations link instances of classes. These relations may be derived from existing relations between their respective classes or gradually discovered.

We classify relations in six categories (c_R): Classification, Meronymic, Similarity, Temporal, Spatial and Any. Some of these categories are characterized by parameters. For instance, in order to define similarity relations between multimedia object, a distance function and a threshold are needed. We thus include the set of attributes A_c in the definition of a relation.

The medical domain has the particularity that a relation can depend on the time. The relation exists in this case in a set of interval of time or in a point of time. For example a *tumor* is *located in* the *breast* during the treatment but after it is no longer there. A patient may also have a recurring tumor. To represent it, we can use a set of time intervals (e.g. Breast cancer, {(12/01/2004-09/04/2006), (04/06/2008-14/02/2011)}). In the relation discovery process, we are also confronted to the problem of uncertainty. The validity of a derived relation may be doubtful. For example in the case of a spatial relation such as "near_of", we can define an uncertainty (the variance) that expresses how near the objects are. Thus, each relation has an optional set of condition properties *cond* that may be composed of temporal and/or uncertainty conditions. A temporal

condition is a set of time intervals and an uncertainty condition is a tuple defining the uncertainty.

2) Categories of relations

In this section, we present relation categories and illustrate them with an example of semantic definition of a relation belonging to the category. In the definitions below, variables c_i ($1 \leq i \leq n$) range over objects and C_i ($1 \leq i \leq n$) range over the corresponding classes. The primitive relation instance_of is represented with the function inst(c, C), which means that c instance of C. We distinguish the following relation categories.

a) Classification: Classification relations capture the semantics of "is_a" relations. They also include the class inclusion relations, which relate two classes when one type subsumes the other (subclass relation).

$$C_1 \text{ is\_a } C_2 := \forall c \, (\text{inst}(c, C_1) \Longrightarrow \text{inst}(c, C_2))$$

Is_a relations can have one of the following two properties:

- The is_a-relation is disjoint, if

$\forall C_1, C_2, C \, (C_1 \text{ is\_a}_d \, C \wedge C_2 \text{ is\_a}_d \, C) \Longrightarrow \nexists c \, (\text{inst}(c, C_1) \wedge \text{inst}(c, C_2))$

- The is_a-relation is total, if

$\forall C_1, C_2, C \, (C_1 \text{ is\_a}_t \, C \wedge C_2 \text{ is\_a}_t \, C) \Longrightarrow \exists c \, (\text{inst}(c, C_1) \vee \text{inst}(c, C_2))$

b) Meronymic: Meronymic relations correspond to "part_of" relations. A part_of relation is a relation of part-hood, where one is a part of the other. For instance, the thorax is a part of the human body.

$C_1 \text{ part\_of } C_2 := \forall c_1 \, \text{inst}(c_1, C_1) \Longrightarrow \exists c_2 \, (\text{inst}(c_2, C_2) \wedge c_1 \text{ part\_of } c_2)$

c) Similarity: Relations in the similarity category express the similarity between two objects. We distinguish between direct similarity and deduced similarity relations. A direct similarity relation represents the similarity between two objects of the same medium type (e.g. two images). The objects must have compatible sets of attributes called feature vectors v_1 and v_2. A similarity score is computed using a specific function $f(v_1, v_2)$ applied on the vectors, which outputs a score *s*. Optionally, the function may require a set of parameters *p*. Two objects are considered similar if their score exceeds a threshold ε. A direct similarity relation thus comprises the following attributes: $A_c = (f, p, s, \varepsilon)$

Direct similarity relations are mostly useful to derive similarity relations between other objects (e.g., two patients are similar because their CT scans have a high similarity score). This type of *derived similarity* relation is defined by a reference to a direct similarity relation.

d) Temporal: We distinguish two types of temporal relations: temporal definition and temporal order. Temporal definition are further divided into "time-interval" relations that correspond to the intuitive notion of time interval and "time-point" relations that correspond to the notion of point in time *[10]*. A time-interval relation has two attributes: begin and end. Temporal order relations are "*after*", "*before*", "*during*", etc. A semantic definition of the relation *during* is for instance the following:

$\forall c_1, c_2 \; c_1$ during $c_2 \Leftrightarrow$ (beginning (c_1) before beginning (c_2)) AND (end (c_2) before end (c_1))

e) Spatial: Spatial relations are used to specify a relation between two objects in the space or to describe a position, which is very important in the medical domain. For instance, the liver lies to the right of the stomach. For instance, the CRC relation located_in is defined as follows.

C_1 located_in $C_2 := \forall c_1 \, \text{inst}(c_1, C_1) \Longrightarrow \exists c_2 \, (\text{inst}(c_2, C_2) \wedge c_1$ located_in $c_2)$

For two objects c_1, c_2 and two regions r_1, r_2 (which are also objects):

c_1 located_in $c_2 := \exists r_1, r_2 \, (c_1$ located_in r_1 AND c_2 located_in r_2 AND r_1 part_of $r_2)$

f) Any: "Any" relations are relations that do not belong to one of the canonical types described above. Among these are relations derived from a foreign key relation between two tables of a relational database. For example, the table Patient of a relational database may have the reference "referring physician name" to the table Physician. This is represented in the dataspace by the CRC relation "has_physician" between the classes Patient and Physician.

D. DATASPACE VIEWS

We introduce in this section dataspace views that we define as sub-dataspaces with data management attributes. They are populated by the results of a query over the dataspace. A dataspace view can be used to conduct a data integration of limited scope, which is restricted to a subset of the data for which the integration is particularly useful. A dataspace view is "virtual" when it is not populated by objects and only comprises classes and relations. On the contrary, a populated dataspace view is "materialized". Dataspace views also have properties with respect to their handling of updates and new data in the sources. They may be "fully-synchronized" (updated on each change in the sources, thus always staying up to date, with very expensive constant updates), "partially-synchronized" (updated at fixed time points), or "non-synchronized" (only synchronized on demand). In practice, a dataspace view is not necessarily populated by data extracted by a query over the dataspace; it can be also partially or totally created from other existing dataspace views. To enable this, a dataspace algebra is being developed. The algebra will define operations like projection, fusion, intersection or join on dataspace views. To outline the advantages of this approach, let us consider an epidemiological study on liver cancer in Bavaria. A materialized dataspace view *V* extracting all available information from the sources could be first created, which would be a very expensive process. Then, all queries required for the study would be directed towards *V*. If the study is restricted to a specific time period, *V* could be created "non-synchronized". Thus no further access to the sources would be required after the creation of *V*.

III. MULTIMEDIA DATASPACE: A REPRESENTATION OF THE SIMILARITY RELATION

In this Section, we outline the representation of a similarity relation between multimedia objects using the model presented in section II. The similarity relation is defined between two mammography images of two patients in a clinic. Patient information is stored in a relational database and the images in the multimedia database. Each image is described in the

multimedia database as an MPEG-7 Document containing color, texture and shape features descriptions. In the dataspace, this information is represented by the classes Patient and Image. Fig. 1 gives an example of a materialized multimedia dataspace view containing this information.

Classes Patient and Image are defined as follows:
Class Patient = (Patient, $cs_{Patient}$, 12/03/2000, {(Name, string), (First name, string), (Birthdate, date), (Sex, char), (Referring Physician, String), (Patient Record Number, integer)}), where $cs_{Patient}$ = (RDB, Data source, http://www.dimis.fim.uni-passau.de/RDB)
Example object: *patient*$_1$ = {(Name, "Nouk"), (First name, "Paula"), (Birthdate, 23/05/1949), (Sex, 'F'), (Referring Physician, "Dr. Njeck"), (Patient Record Number, 1295645)}

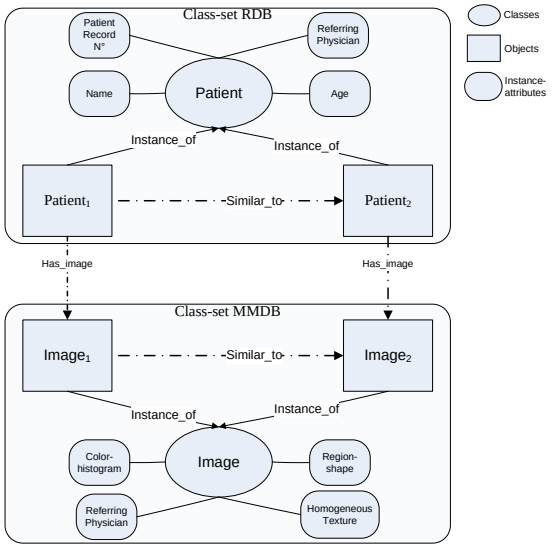

Fig. 1 Classes, objects and relations in a multimedia dataspace

Class Image = (Image, cs_{Image}, 23/05/2000, {(Patient record Number, integer), (Doc_ID, integer), (HomogeneousTexture, integer), (ScalableColor, integer), (RegionShape, integer), (FreeText, string)}), where cs_{Image} = (MMDB, Data source, http://www.dimis.fim.uni-passau.de/MMDB)
Example object*: image*$_1$ = {(Doc_ID, 23), (Patient Number, 1295645), (mpeg7:HomogeneousTextureType, v_H), (mpeg7:ScalableColorType, v_S), (mpeg7:RegionShapeType, v_R).

As we have images corresponding to patients, we can define a direct similarity relation between the images objects and consequently derive the similarity between the Patients. The features vectors of the image objects are (HomogeneousTexture, ScalableColor, RegionShape). We define the relations of the dataspace as follows:

is_similar_image = (is_similar_image, $image_1$, $image_2$, p_{isi}), where p_{isi} = (ORO, IRelation, Similarity, http://www.dimis.fim.uni-passau.de/image/similarity/ORO, {ε = 1, dist:EuclidianType})

is_similar_patient = (is_similar_patient, $patient_1$, $patient_2$, p_{isp}), where $p_{isp\_}$= (ORO, IRelation, Similarity, http://www.dimis.fim.uni-passau.de/image/similarity/CRC)
has_image = (has_image, $patient_1$, $image_1$, p_{has_image}), where p_{has_image}= (ORC, ERelation, Any, http://www.dimis.fim.uni-passau.de/image/similarity/ORC)

We assume the use of namespaces to define the controlled vocabulary used in the dataspace. Thus, "dist:EuclidianType" uniquely identifies the Euclidian distance, and the "mpeg-7:xx" fields identify the corresponding MPEG-7 descriptors.

IV. Conclusion and Future Work

In this paper, we have presented a dataspace model, which can represent many types of relations and includes features specific to the multimedia field. In the model a dataspace is defined as a set of entities (classes/objects) and relations between the entities. Classes and objects can be extracted from data sources, ontologies or canonical knowledge. The model also introduces dataspace views, which can be used to create and manipulate sub-dataspaces in a flexible way. Future work includes the formalization of an algebra operations like projection, intersection or union applied to dataspace views. Moreover, we intend to implement the model using RDF and to develop prototypes for creating dataspaces and dataspace views from different types of data sources.

References

[1] Gantz, J., Reinsel, D., Chute, C., Schlichting, W., Mcarthur, J., Minton, S., Xheneti, I., Toncheva, A., Manfrediz, A.: IDC - The Expanding Digital Universe: A Forecast of Worldwide Information Growth Through 2010., EMC (2007)

[2] Halevy, A.: Why Your Data Won't Mix. Queue 3(8), 50-58 (2005)

[3] Franklin, M.: From databases to dataspaces: A new abstraction for information management. SIGMOD Record 34, 27-33 (2005)

[4] Halevy, A., Franklin, M., Maier, D.: Principles of Dataspace Systems. PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 1-9 (2006)

[5] Jeffery, S., Franklin, M., Halevy, A.: Pay-as-you-go user feedback for dataspace systems. In : SIGMOD Conference. (2008) 847-860

[6] Dittrich, J.-P., Blunschi, L., Färber, M., Girard, O., Karakashian, S., Salles, M.: From Personal Desktops to Personal Dataspaces: A Report on Building the iMeMex Personal Dataspace Management System. In : BTW 103. GI (2007) 292-308

[7] Sarma, A., Dong, X., Halevy, A.: Data Modeling in Dataspace Support Platforms. In : Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos. Springer-Verlag (2009) 122-138

[8] Vaz Salles, M., Dittrich, J.-P., Karakashian, S., Girard, O., Blunschi, L.: iTrails: pay-as-you-go information integration in dataspaces. In : VLDB '07: Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, Vienna, Austria (2007) 663-674

[9] Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space 1st edn. I. Morgan & Claypool (2011)

[10] Benjamin, P., Menzel, C., Mayer, R., Fillion, F., Futrell, M., deWitte, P., Lingineni, M.: IDEF5 Method Report., Knowledge Based Systems, Inc. (1994)