DOI: 10.1515/awutm -2016-0011

DE GRUYTER OPEN

Analele Universităţii de Vest,
Timişoara
Seria Matematică – Informatică
LIV, 2, (2016), 3– 11

Towards Building a Uniform Cloud Database Representation for Data Interchange

Alina Andreica

Abstract. The paper proposes design principles for data representation and simplification in order to design cloud services for data exchange between various information systems. We use equivalence algorithms and canonical representation in the cloud database. The solution we describe brings important advantages in organizational / entity communication and cooperation, with important societal benefits and can be provided within cloud architectures. The generic design principles we apply bring important advantages in the design of the interchange services.

AMS Subject Classification (2000). 68P05 ; 68P20 ; 68W30
Keywords. equivalence and simplification algorithms, data interchange, cloud services, pattern matching, software design

1 Introduction and Working Framework

Software design principles apply systematic techniques to application design, use abstract patterns for process modeling and implement adequate tools for problem solving. Within this framework, we propose means of using simplification and equivalence algorithms for modelling data representation in order to ensure data interchange between information systems by means of cloud services. Equivalence algorithms can be implemented in an abstract manner, based on category theory [1]. Applying generic techniques is useful both for design reasons and for tackling specific problems based on certain mathematical models [1]. Interoperability is the capability of different systems to share functionalities or data [2]. System interoperability [3] has been dealt with by means of various models [3] and has been extensively researched for business processes [5]. In [6] we overview interoperability layers, principles and tools. Ones of the most relevant are: The Electronic Data Interchange (EDI) model [7], the XML standard [8], RosettaNet [9], ebXML [10] standards. Knowledge discovery, inference, logic are enabled by semantic interoperability. Knowledge sharing over computer information systems is a major task of the interoperability approach and is based on the Conceptual Knowledge Processing paradigm [11]. The Open Internet of Things standards [12] may also be used as an efficient framework for data interchange. Within section 2 we address means of implementing simplification and equivalence algorithms on various entities, including hierarchical structures. Section 3 describes the way in which specific pattern matching problems can be solved by using equivalence algorithms. Section 4 addresses principles of data interchange between information systems using a cloud database retaining data in a canonical representation. Conclusions reveal the most important topics presented in the paper and future research and development directions.

2 Equivalence Algorithms

Within this section we present means of implementing equivalence algorithms [13] on various entities, including hierarchical structures. We use the implementation framework that we have introduced in [1]. An equivalence relation $' \approx '$ verifies reflexivity, symmetry, transitivity properties [13]. Since the data volume that has to be processed is usually large, the most efficient way of organizing it is using a relational database, in which entities are retained in dedicated tables. Database structuring principles for processing equivalent entities are introduced in [1]. We use as well tables for retaining the entities on which equivalence algorithms are applied. We process entities belonging to the dedicated tables that retain those entities:

$$IsSpecificEntity(d) := \begin{cases} \textit{true}, d \in \textit{Tbl\_entity}[\textit{id\_entity}] \\ \textit{false}, \textit{otherwise} \end{cases}$$

Hierarchical data structures are often necessary to be processed in a database; such structures may be retained in relational databases by means of ascendant / successor pointers in dedicated tables. Principles for retaining and processing hierarchical structures and a comparison of their processing techniques are presented in [4].We note that such a hierarchical structure can often be encountered and therefore is useful to be processed. In [1] we present means of processing hierarchical structures at database level, using a dedicated table for retaining the corresponding entities and a 4 pointers retaining technique: ascendant, descendant, predecessor (same level), successor (same level). In [1] we present the case study of equivalent disciplines and in [4] an example for plant therapy.

In [4] we detail these principles for processing modules of didactic activities. The implementation uses stored procedures parameterized with the level value, data selection operations being performed dynamically, in respect with this value. The system uses a MS SQL database [9] and the hierarchical structures which model curricula information are mainly processed by means of stored procedures (details can be found in [4]).

Postorder type n-ary tree evaluation algorithms based on the above described tree representation are implemented in order to parse the hierarchy of entities. Some efficiency studies we have performed on processing hierarchical structures at database level are given in [15]. In a general hierarchical entity structure, leaf entities are the ones retained in the basic dedicated tables.

$$IsLeafEntity(m) := \begin{cases} \textit{true}, \forall d \in m : \textit{IsSpecificEntity[d]} \\ \textit{false, otherwise} \end{cases}$$

For example, in [1], we process hierarchies of modules in which all nonleaf modules consist only of modules. Let $d1$, $d2$ be two entities. We use the notation $'\sim'$ for describing the equivalence of the two entities $d1 \sim d2$; this relation may have various significances in various case studies see [4], [1]. In each case, we have to check whether the relation is an equivalence one since by verifying if it complies reflexivity, symmetry, transitivity properties. The canonical representative of an entity equivalence class is important since it will be further used in pattern matching rules see section 3. We implemented the simplification algorithm for determining the canonical representative [6] for a given entity. Based on this algorithm, we may also test the equivalence of two entities by verifying they have the same canonical representative. By generically denoting with $'\approx'$ an equivalence relation for categories of entities, we may state that:

$$e1 \approx e2 \Leftrightarrow (\forall d1 \in e1, \exists! d2 \in e2 : d1 \sim d2) \wedge$$

$$(\forall d2 \in e2, \exists! d1 \in e1 : d1 \sim d2)$$

For a leaf category of entities e, we consider Canonic$(e)=\{$Canonic$(d)|d\in e\}$ the set of canonical representative for the contained entities. It can be shown that two leaf equivalent entities have the same sets of canonical representatives. For a category of entities we can recursively compute its canonical representative set as:

$$\text{Canonic}(e)=\begin{cases}\{\text{Canonic}(d)|d\in e\}, \textit{IsLeafEntity[e]}\\ \{\text{Canonic}(ed)|ed\subset e\}, \textit{otherwise}\end{cases}$$

Intuitively, the canonical set for a category of entities is obtained by "flattening" its category sub-tree and computing the union set of all canonical sets corresponding to its descendant leaf entities. Generically, we may state:

$$\text{Canonic}(e)=\{\text{Canonic}(d)|d\in e\}$$

3 Pattern Matching Principles

We implement pattern matching rules for equivalent entities by reducing the mapping between two elements, belonging to the two equivalence classes that are to be mapped, to mapping their canonical representatives, as described below: Let $e_i\in E$ where E is a class of equivalent entities, $em_j\in EM$ where EM is a class of equivalent mapped entities, $e0$ the canonical representative of class E and $em0$ the canonical representative of class EM. Then we reduce a mapping of two entities e_i, em_j belonging respectively to the equivalence classes E, EM to the mapping between the two canonical representatives $e_0\in E, em_0\in EM$ see Figure 1:

$e_i\to em_j\to e_0\to em_0$, where $e_i\in E, em_j\in EM$.

We may as well use the equivalent mappings: $e_i\to em_0$ or $e_0\to em_j$, where $e_i\in E, em_j\in EM$.

For the case of equivalence classes with hierarchical representations see Figure 2 the canonical representatives are the roots of the corresponding trees (Figure 2). Parsing algorithms for finding the canonical representatives generally use the ascendant pointer see section 2. We can use the above described rules in managing pattern matching problems on equivalence classes that occur in the design of expert systems.

We note that for the database representation it is important to improve table access speed table by indexing the tables in respect with the search id, this principle is very useful to be applied as well in managing hierarchical representations at database level, which are frequently processed in order to find the canonical representative.

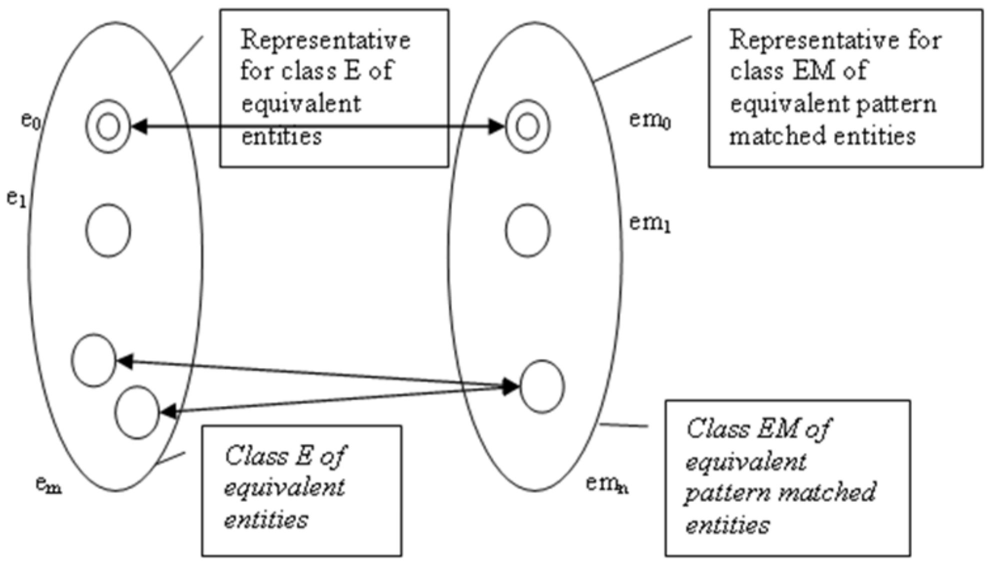
Representative for class E of equivalent entities
Representative for class EM of equivalent pattern matched entities
e_0
e_1
e_m
em_0
em_1
em_n
Class E of equivalent entities
Class EM of equivalent pattern matched entities

Figure 1: Pattern Matching scheme for Equivalence classes

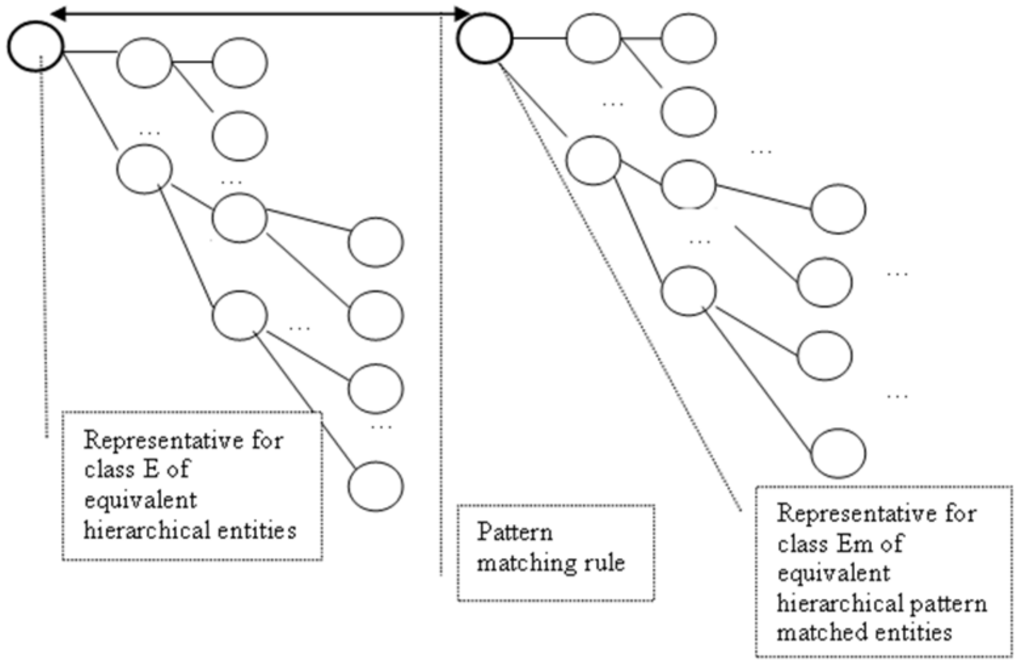
Representative for class E of equivalent hierarchical entities
Pattern matching rule
Representative for class Em of equivalent hierarchical pattern matched entities

Figure 2: Pattern Matching on Hierarchical Structures of Equivalence Classes

4 Principles for Building the Cloud Uniform Database Representation

In order to enable the exchange of various data between two information systems, using dedicated databases, with different structures, we will set the canonical representation on the cloud database. The mapping process will be user assisted since it requires human input. The data interchange architecture [6] provides data exchange services between various information systems or entities using cloud services and a cloud database for mapping and handling the exchanged information see Figure 3. Data exchange may be performed both in XML relational database formats; for XML format, the corresponding database representation [1] is generated into the cloud database. The following sequences are pursued: data to be exchanged is marked in the source database using dedicated columns, mapped into the cloud database, sent and retained into the cloud database. For the destination system / database, which sends data requests, a data mapping is also performed and required data is sent from the cloud database into the destination one. The data exchange may use multi-criteria agents implemented in the cloud environment both for performing necessary mappings and for handing communication. The cloud data interchange services are to be designed for supporting automatic data exchange in various fields, with important communication efficiency benefits [6].

5 Conclusion and Future Work

The paper proposes data representation and design principles for performing data exchange between various information systems and databases by means of cloud services. Simplification and equivalence algorithms are used for ensuring canonical data representation in the cloud database and ensuring data correspondence in the data exchange process. The generic manner in which we implement simplification and equivalence algorithms on various entities, including hierarchical ones, represented at database level, ensures generality and applicability in various cases. Pattern matching rules and canonical representatives are used in the cloud database. We reveal the advantages of applying the algebraic equivalence model and of applying canonical representatives properties in solving pattern matching problems and designing

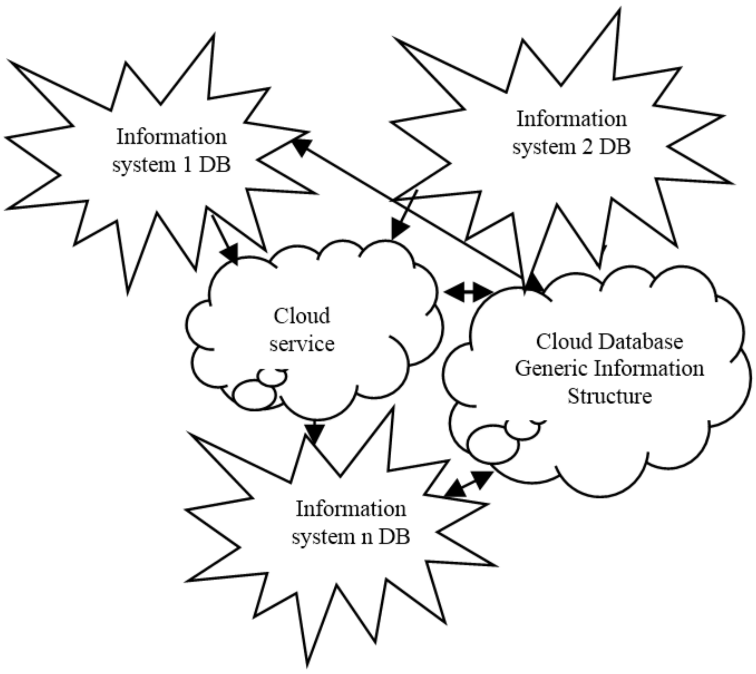

Figure 3: Data Interchange Model Using Cloud Services

data exchange services. The data interchange model we present provides important practical advantages for increasing organizational competitiveness, with a significant societal impact on institutional and entities cooperation, efficient information access and management for various stakeholders. A relevant advantage of the solution is its flexibility and efficiency in information exchange (only relevant data is exchanged), with minimal resources involved. The model we describe develops a standardization framework for enabling data interchange, supports information management and data exchange, using cloud services and ensuring systems and entities interoperability. Future work is related to further development and implementation of the above described principles.

References

[1] **Alina Andreica, Daniel Stuparu, and Calin Miu**, *Applying Mathematical Models in Software Design*, 2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, Proceedings of ICCP 2012, IEEE, Ed: Ioan Alfred Letia, 2012, 87–90

[2] **Daniel Olmedilla, Nobuo Saito, and Bernd Simon**, Educational Technology & Society, *Special Issue on Interoperability of Educational Systems*, **9**, (2006)

[3] **Edwin Morris, Linda Levine, Craig Meyers, Pat Place, and Dan Plakosh**, System of Systems Interoperability (SOSI): Final Report, *http://www.sei.cmu.edu/reports/04tr004.pdf , retrieved May 2016*, (2004)

[4] **Alina Andreica, Daniel Stuparu, and Calin Miu**, *Design Techniques in Processing Hierarchical Structures at Database Level*, Proceedings of Iadis Information Systems 2010, Porto, 18-20 March 2010, IADIS Press, Ed: M Nunes, P Isaias, P Powell, 2010, 483–488

[5] **Jörg Ziemann**, *Architecture of Interoperable Information Systems - An Enterprise Model-Based for Describing and Enacting Collaborative Business Processes*, Logos Verlag, Berlin, 2010

[6] **Alina Andreica, Florina Covaci, and Josef Küng**, *A Generic Model for Cloud Data Interchange*, Proceedings of 14h RoEduNet International Conference - IEEE, Craiova, 24-26 September 2015, IEEE Computer Society, 2015, 138–142

[7] **Susie Adams, Dilip Hardas, Aktar Iossein, and Charles Kaiman**, *BizTalk Unleashed*, Sams Publishing, p. 966, Indianapolis, Indiana, 2002

[8] *******, XML standard, *http://www.w3.org/TR/xml11/#charsets, retrieved November 2015*

[9] *******, Rosetta Rosettanet Overview: Clusters, Segments, and PIPs (ver 02.13.00), *http://www.rosettanet.org/TheStandards/ RosettaNetStandards/PIPOverview/tabid/3482/Default.aspx, retrieved December 2015*, (2011)

[10] *******, OASIS OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features, *http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms_core-3.0-spec.pdf, retrieved December 2015*, (2007)

[11] **Gerd Stumme and Rudolf Wille**, *Begriffliche Wissensverarbeitung / Conceptual Knowledge Processing*, Springer Verlag, 2000

[12] **Bruno Buchberger and Loos Rudiger**, OpenIoT - Open Internet of Things architecture, *Algebraic Simplification, Computing*, **Suppl. 4**, (https://github.com/OpenIotOrg/openiot/wiki/OpenIoT-Architecture, retrieved December 2014B.), 11-43

[13] **Alina Andreica**, Applying Equivalence Algorithms in Solving Pattern Matching Problems. Case Study for Expert System Design, *Proceedings of the international Conference in on Theory and Practice in Modern Computing TPMC 2016, July 1-4, Portugal*, (2016), 255–259

[14] **Alina Andreica, Daniel Stuparu, and Calin Miu**, Design Techniques in Processing Hierarchical Structures at Database Level, *Proceedings of Iadis Information Systems 2010, Porto, 18-20 March 2010, Ed: M Nunes, P Isaias, P Powell*, (2010), 483–488

[15] **Alina Andreica, Daniel Stuparu, and Iulia Mantu**, Symbolic Modelling of Database Representations, *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing 2005*, (2005), 59–62

Alina Andreica
Faculty of European Studies
Babes-Bolyai University
1, Kogalniceanu Street
Cluj-Napoca
Romania
E-mail: `alina.andreica@ubbcluj.ro`

Received: 23.07.2016
Accepted: 6.11.2016