A Universal Ontology-based Approach to Data Integration

Antoni Olivé*,[a]

[a] Universitat Politècnica de Catalunya – BarcelonaTech. Catalonia

Abstract. *One of the main problems in building data integration systems is that of semantic integration. It has been acknowledged that the problem would not exist if all systems were developed using the same global schema, but so far, such global schema has been considered unfeasible in practice. However, in our previous work, we have argued that given the current state-of-the-art, a global schema may be feasible now, and we have put forward a vision of a Universal Ontology (UO) that may be desirable, feasible, and viable. One of the reasons why the UO may be desirable is that it might solve the semantic integration problem. The objective of this paper is to show that indeed the UO could solve, or at least greatly alleviate, the semantic integration problem. We do so by presenting an approach to semantic integration based on the UO that requires much less effort than other approaches.*

Keywords. Universal Ontology • Data Integration • Mediated Schema • Conceptual Modelling.

1 Introduction

The goal of data integration systems is to offer uniform access to a set of autonomous and heterogeneous data sources. Building such systems is difficult for system, logical and social reasons (Doan et al. 2012, p. 6). One of the main problems is that of semantic integration, which arises from the fact that the conceptual schemas of the data sources to be integrated have been developed independently. For some authors, despite its pervasiveness and importance, semantic integration remains an open and extremely difficult problem (Batini et al. 1992, Park and Ram 2004).

Several authors have acknowledged in the past that the semantic integration problem would not exist, or at least it would be greatly alleviated, if all systems were developed using the same global schema (Madnick 1996, Uschold 2000, Mena et al. 2000, Grüninger and Uschold 2004). However, such global schema has been considered unfeasible in practice, and therefore, as far as we know, data integration in the context of a global schema has not been explored in the same level of detail as in other contexts.

In a recent paper (Olivé 2017), we have argued that a global schema may have been unfeasible in the past, but it is no longer the case now. We have put forward a vision of a global schema, called the Universal Ontology (UO), that is desirable, feasible in the current state of the art, and viable.

One of the reasons why the UO is desirable is that it might solve the semantic integration problem in data integration systems. However, as we have said, little is known about the use of the UO in those systems and its effectiveness in solving that problem. We review some of the work that has been done in the section on related work.

The objective of this paper is to show that indeed the UO solves, or at least greatly alleviates, the semantic integration problem. We do so by presenting an approach to data integration based on the UO that requires much less effort than other approaches. The approach requires that the developers of the data sources define the semantic mappings between the data source schemas and

* Corresponding author.
E-mail. antoni.olive@upc.edu
This work has been partially supported by the Ministerio de Economía y Competitividad, under project TIN2014-52938-C2-2-R.

the UO. From these semantic mappings, the other semantic components of a data integration system (mediated schema, source descriptions) can in principle be automatically generated.

The rest of the paper is structured as follows. Next section briefly describes the scope, the kinds of the concepts, the elements that comprise the specification of each concept of the UO, and the concept composition operators. Section 3 explains how to define the mappings between data sources and the UO. Section 4 presents the framework of the data integration system, and the semantic components to be generated. Section 5 presents a method for the generation of the mediated schema and the source descriptions. Section 6 discusses some related work. Finally, section 7 summarizes the conclusions. Throughout the paper we use the example introduced in (Doan et al. 2012, p. 67) with minor modifications.

This paper has been written as a sincere tribute to Heinrich C. Mayr on occasion of his retirement. During his professional life as researcher, he has made substantial contributions to the conceptual modelling field since the beginning (Lockemann et al. 1979) until recently (Michael and Mayr 2017) and, at the same time, he has had an active involvement in many organizational activities that have helped to build a strong and friendly conceptual modelling community.

2 The Universal Ontology

In this section, we summarize the characteristics of the UO proposed in Olivé (2017) that we will need in this paper. We explain first the specification of the UO concepts and then the concept composition operators.

2.1 Concept specification

The objective of the UO is to allow the publication, search, and reading by people and machines of any fact of any domain, using an integrated set of all existing concepts. The UO comprises three kinds of concepts: entity types, datatypes and binary properties. There are two kind of properties: object properties, which link entities to entities, and datatype properties, which link entities to data values. Properties have a direction, from subject (domain) to object (range).

We have argued that WordNet (Miller 1995), among others, could be the basis of a substantial part of the UO. For this reason, the examples of this paper are taken from WordNet, although we make an ad hoc use of datatypes.

The specification of each concept includes at least:

- The kind of the concept.
- The concept identifier. Each concept should have a natural language-independent, unique, and immutable identifier. In the examples of this paper, we will use as concept identifier the word# sense number of the concept in WordNet, such as the noun *Movie#1*.
- The name and synonym(s) of the concept in each natural language spoken by the UO users. The names of the concepts need not be unique. In general, the name of an entity or data type must be a noun phrase, while the name of a property must be a verb phrase or a noun phrase. When the name of a property is a noun phrase, the property is seen as an attribute (characteristic, feature . . .) of the subject. Most of the properties in the examples of this paper are attributes. We will assume that their identifier has the general form *HasType*, where *Type* is the identifier of an entity or datatype. For example, *HasTitle#2*.
- The definition of the concept. It may be a natural language description or a derivation rule in some formal language.
- The supertypes of the concept (*IsA* relationships).
- The analytical constraints that the instances of the concept must satisfy to be considered universally valid. Among these constraints there are the allowed domain and range of properties, and the disjointness constraints of concepts.
- The (meta-)entity types of which an entity type is an instance (*InstanceOf* relationships).

2.2 Concept composition

The UO described above specifies only a limited (even if very large) number of concepts. However, it is a fact that using an appropriate set of composition operators we could compose a limitless number of concepts from them. We call core UO the explicitly defined ontology, and extended UO the set of concepts that could be composed from the core. The full UO would then be the union of the core and extended parts.

In the examples of this paper, we will use only the reification operator described in the following. Other operators have been defined in Olivé (2017).

The reification operator is well known in conceptual modelling. Let P be a property, E_1 an entity type that is in the domain of P and E_2 an entity or data type that is in the range of P. Then, E = $Reif(E_1,P,E_2)$ is an entity type that corresponds to the reification of P with domain E_1 and range E_2. An instance of E corresponds to an instance of P such that its subject is an instance of E_1 and its object is an instance of E_2.

The UO includes two properties that can be used when needed:

- *HasSubject*, which links a reification E to its subject E_1.
- *HasObject,* which links a reification E to its object E_2.

3 Mapping relational schemas to the UO

Our approach to data integration requires the mapping of the data sources to the UO. This mapping is done for each data source, independently of other data sources with which it might be integrated. In fact, it might be justified to do this mapping as a means for documenting a data source schema.

In this paper, we will focus only on data sources that are relational databases.

3.1 Anchors

Each relational schema R must be mapped to an entity type E of the full UO, which we call the anchor of R, written as *anchor*(R) = E (An et al. 2006). The meaning is that a tuple of R represents data about an instance of E. Each tuple of R corresponds to a different instance of E. In a given database, there may be several relational schemas whose anchor is the same entity type.

In the example, there are six relational databases, S_1 to S_6. The relational schemas of these databases and their corresponding anchors are the following:

S_1:
  R_1: *Movie*(*MID*,*title*) *anchor*(R_1) = *Movie#1*
  R_2: *Actor*(*AID*,*firstName*,*lastName*,*nationality*, *yearOfBirth*) *anchor*(R_2) = *Actor#1*
  R_3: *ActorPlays*(*AID,MID*)
  anchor(R_3) =
  Reif(*Movie#1,HasActor#1,Actor#1*)
  R_4: *MovieDetails*(*MID*,*director*,*genre*,*year*)
  anchor(R_4) = *Movie#1*
S_2:
  R_5: *Cinemas*(*place,movie,start*)
  anchor(R_5) = *Show#3*
S_3:
  R_6: *Reviews*(*title,date,grade,review*)
  anchor(R_6) = *Review#2*
S_4:
  R_7: *MovieGenres*(*title,genre*)
  anchor(R_7) = *Movie#1*
S_5:
  R_8: *MovieDirectors*(*title,dir*)
  anchor(R_8) = *Movie#1*
S_6:
  R_9: *MovieYears*(*title,year*)
  anchor(R_9) = *Movie#1*

Note that, in the example, the anchor of most relational schemas is *Movie#1*. The anchor of R_2 is *Actor#1*. The anchor of R_3 is an entity type of the extended UO. In this case, it is the result of the reification operator, introduced in the previous section. Each tuple of R_3 corresponds to a participation of an actor in a movie. The anchor of R_5 is *Show#3* (”a social event involving a public performance or entertainment”). The anchor of R_6 is *Review#2* (”an essay or article that gives a critical evaluation (as of a book or play)”).

3.2 Mapping attributes

In a relational schema R, each attribute A of type T maps to a datatype property P of the full UO. The domain of P for A in R is the same of P in the UO or a subtype of it. The range of P for A in R is T, which must be the same of P or a subtype of it.

We write *map*(A) = P(D,Rg) to indicate that attribute A maps to property P and that the domain of P in R for A is D, and its range Rg. For example, the mapping of attribute *title* of R_1 of type *String* is:

map(*title*) = *HasTitle#2*(*Movie#1*,*String*)

This means that attribute *title* of R_1represents the value of property *HasTitle#2* of the instances of *Movie#1*, which is the anchor of R_1. The mapping would be the same for attribute *title* in R_7, R_8 and R_9.

For mapping purposes, we must distinguish two kinds of attributes in R, which we call direct and indirect. A direct attribute A is an attribute of the anchor of R, and, therefore, it maps to a data type property P whose domain is the anchor of R and its range is T.

An example is the attribute *title* as indicated above. Another example may be the attribute *start* of R_5 whose type is assumed to be *Time*. It maps to the datatype property *HasShowTime#1*:

map(*start*) = *HasShowTime#1*(*Show#3*,*Time*)

An indirect attribute A of type T of R is an attribute of an entity type O related to the anchor of R by means of an object property P_1. O is called an indirect entity type of R. In this case, we write:

map(A) = (P_1(*anchor*(R),O),P(O,T))

For example, in R_4, attribute *director* is the name of the director of a movie, that is:

map(*director*) =
(*HasDirector#4*(*Movie#1*,*Director#4*),
HasName#1(*Director#4*,*String*))

In this case, an indirect entity type of R_4 is *Director#4*. Note that *HasName#1* refers to the name of a director, not that of the anchor of R_4 (movie). The same mapping applies to attribute *dir* of R_8.

As another example, we have attribute *year* of R_4 and R_9, which refers to the year in which a movie was released. Both map to:

map(*year*) =
(*HasRelease#2*(*Movie#1*,*Release#2*),
HasDate#1(*Release#2*,*Year*))

In R_5 we find two indirect attributes:

map(*place*) =
(*HasCinema#2*(*Show#3*,*Cinema#2*),
HasName#1(*Cinema#2*,*String*))
map(*movie*) =
(*Show#1*(*Show#3*, *Movie#1*),
HasTitle#2(*Movie#1*,*String*))

When the anchor of R is a reification, then R has one or more attributes that map to the subject, and one or more attributes that map to the object of the reification.

In the example, for R_3 we have

map(*MID*) =
(*HasSubject*(E,*Movie#1*),
HasIdentifier#1(*Movie#1*,*Integer*))
map(*AID*) =
(*HasObject*(E,*Actor#1*),
HasIdentifier#1(*Actor#1*,*Integer*))

where E = *Reif*(*Movie#1,actor#1,Actor#1*). Note that in this case the two attributes are indirect.

3.3 Identifiers

Each relational schema must have at least one identifier of its anchor, and may have one identifier of each indirect entity type, if any. An identifier consists of one or more attributes whose values identify the corresponding instances. An identifier is simple if it consists of only one attribute, and composite if otherwise.

For example, the anchor of R_1 has two identifiers, both simple. The first is the attribute *MID*, and the second is the attribute *title*. In R_2, the anchor has two identifiers, one simple and the other composite. The simple is attribute *AID*. The composite consists of attributes *firstName* and *lastName*. In R_4, the indirect object *Director#4* has a simple identifier (attribute *director*).

In data integration, it is important to distinguish between local and global identifier attributes, depending on whether they are locally or globally known. The value of a local identifier attribute identifies an entity in a way that is known only in the context of a data source, while that of a global one identifies an entity in a way that is or may be known globally.

For example, in R_1, *MID* is local, while *title* is global. The values of *MID* identify movies in a way that is known only in S_1, while the values of *title* identify movies in all data sources.

As another example, the anchor of R_6 has a composite identifier, consisting of two global identifier attributes *title* and *review*, whose mappings are:

map(*title*) =
(*Review#2*(*Review#2*,*Movie#1*),
HasTitle#2(*Movie#1*,*String*))
map(*review*) = *HasText#1*(*Review#2*,*String*)

Note that in the above example, *title* is an indirect attribute whose indirect entity type is *Movie#1*. The indirect object property is written as *Review#2*, which in this case is the WordNet verb synset *review#2* (" appraise critically").

When the anchor of a relational schema *R* is a reification, then *anchor*(*R*) has normally a composite identifier consisting of two attributes: one for the subject and one for the object of the reification. In the example of R_3, *anchor*(R_3) has a composite identifier, consisting of the two local identifier attributes *MID* and *AID*.

3.4 Assessment

In general, designers and users of relational databases know the anchor and the properties of their relational schemas. As we have seen, our approach requires that anchors and properties are related to the corresponding concepts in the UO. This should be easy if the concepts are in the core part, and may be not so easy if they must be composed. Anyway, this must be done only once per relational schema and, on the other hand, it is useful as a documentation of the semantics of the schema. We assume that the UO will be large enough to include most of the possible anchors and properties. When this is not the case, the corresponding relational schema will need to be manually integrated.

4 Data integration framework

In this section, we briefly introduce the data integration framework in which we place our work. We take as a basis the work reported in Lenzerini (2002) and Doan et al. (2012).

The goal of a data integration system is to combine the data residing at different sources, and to provide the users with a unified view of these data. The unified view is represented by the mediated schema, which is a reconciled view of all data that can be queried by the user. The main components of a data integration system are the mediated schema, the data sources, and the mappings of the data sources to the mediated schema, also called source descriptions.

There are three main approaches for specifying the mappings: Local-as-View (LAV), Global-as-View (GAV) and Global-and-Local-as-View (GLAV). In our approach we use LAV mappings. Such mappings associate each element of a data source to the mediated schema, independently of any other data sources.

The main tasks in the design of a data integration system are to design the mediated schema and to define the mappings between the sources and the mediated schema. In the next section, we describe how these tasks can be performed when the data sources are relational databases whose schemas have been mapped to the UO as indicated in the previous section.

5 The mediated schema and source descriptions

When the data source schemas are mapped to the UO as indicated in section 3, the mediated schema and the source descriptions can be obtained as indicated in the following. We explain first the mediated schema and then the source descriptions.

5.1 Mediated schema

In our approach, we assume that the mediated schema must include all entity types, datatypes and properties represented in the local data sources.

In the example (Doan et al. 2012, p. 67), the mediated schema is defined by the following four relation schemas:

M_1: *Movie*(*title*,*director*,*year*,*genre*)
M_2: *Actors*(*title*,*name*)
M_3: *Plays*(*movie*,*location*, *startTime*)
M_4: *Reviews*(*title*,*rating*,*description*)

Note that the R_2 attributes *nationality* and *yearOfBirth* are not included in the mediated schema. The same happens with the local identifier attributes *MID* and *AID* in S_1, and the attribute *date* of R_6.

For simplicity's sake, and without loss of generality, we define the mediated schema using the logic formalism (An et al. 2006, p. 6); (Olivé 2007, ch. 2,3). Entity types and data types are represented by means of unary predicates, while properties are defined by means of binary predicates. *IsA* relationships, constraints and queries will be represented by first-order logic expressions.

The mediated schema M consists of a set of entity and data types, with their *IsA* relationships and disjointness constraints, and a set of properties, with their domain and range constraints. We deal first with the entity types and then with the properties. Data types are treated similarly to entity types.

Entity types. The entity types of M are the set of anchors and indirect entity types of all relational schemas of all data sources, and their entailed types, if any.

The *IsA* relationships of M are those of the entailed types and those defined in the UO. If E_1 and E_2 are two entity types in M and there is in the UO a direct or indirect E_1 *IsA* E_2 then there must be also a direct or indirect E_1 *IsA* E_2 in M.

The disjointness constraints of M are those defined in the UO. If E_1 and E_2 are two entity types in M and there is in the UO a direct or indirect disjointness between E_1 and E_2 then there must be also a direct or indirect disjointness between E_1 and E_2 in M.

Using our approach, the entity types of the mediated schema corresponding to the anchors are: *Movie#1, Actor#1, Reif (Movie#1,HasActor#1,Actor#1), Show#3*, and *Review#2*, and those corresponding to the indirect entity types are: *Country#1, Director#4, Cinema#2* and *Release#2*. In this example, we will ignore the *IsA* relationships and the disjointness constraints.

Properties. The properties of M are the set of mapped properties of all relational schemas of all data sources, and their entailed properties, if any. The *IsA* relationships and the disjointness constraints of the properties, which is this example we will ignore as before, are represented similarly to those of the entity types.

Different attributes of the relational schemas of the data sources may map to the same property P of the UO, with the same or different pairs of domain D and range Rg. A pair D, Rg of P is called a realization of P (Olivé 2007, ch. 7).

Each realization is represented in the mediated schema by a distinct predicate. For simplicity, we will name $P[D,Rg]$ the binary predicate corresponding to property P with domain D and range Rg. For example:

HasTitle#2 [*Movie#1, String*]
HasFirstName#1 [*Actor#1, String*]
HasLastName#1 [*Actor#1, String*]

There is one exception to the above rule, which concerns local identifier attributes like *MID* in R_1, R_3 and R_4, whose mapping is:

map(*MID*) = *HasIdentifier#1*(*Movie#1*,*Integer*)

This attribute cannot be represented in the mediated schema by the binary predicate *HasIdentifier#1* [*Movie#1, Integer*] because it can be used only in data source S_1. The solution we propose, inspired in (Fowler 1997, p. 88), UN/CEFACT (2009) and SAP (2016), consists in using in these cases a quaternary predicate, which may have the same name as before. The meaning of an atomic sentence such as

HasIdentifier#1[*Movie#1*,*Integer*](*m*,*i*,*a*,*s*)

is that movie *m* has identifier *i* according to the rules defined by the agency *a* in the identification scheme *s*.

5.2 Source descriptions

Once we have obtained the mediated schema, we can automatically generate the source descriptions. We will use LAV mappings.

The general form of a LAV mapping will be:

$R(X) \rightarrow \exists\,!y\;(E(y) \wedge \phi(X,y))$

where R is a relational schema, X the set of attributes of R, E the anchor of R, and $\phi(X,y)$ a formula over the predicates of the mediated schema. The variables in X are implicitly universally quantified in front of the formula. The mapping states that each tuple of R with attributes X maps to one and only one instance y of its anchor E for which $\phi(X,y)$ holds.

For example, the source description of R_7 is:

MovieGenres(*title,genre*) → ∃ !*y*(*Movie#1*(*y*) ∧
HasTitle#2[*Movie#1,String*](*y*,*title*) ∧
HasGenre#1[*Movie#1,String*](*y*,*genre*))

Each direct attribute A of type T for which *map*(A) = $P(D,Rg)$ has the atom $P[D,Rg](y,t)$ in ϕ. This expresses that if t is the value of attribute A in a tuple then $P[D,Rg](y,t)$ must be true. In the above example, we have:

map(*title*) = *HasTitle#2* (*Movie#1*,*String*)

and therefore

HasTitle#2[*Movie#1,String*](*y*,*title*) must be true.

When an attribute is a local identifier, such as *MID* in R_1, R_3 and R_4, the corresponding atom is a quaternary predicate as explained above. In this way, the source description of R_1 is:

Movie(*mid*,*title*) → ∃ !*y*(*Movie#1*(*y*) ∧
HasIdentifier#1[*Movie#1*,*Integer*]
(*y*,*mid*,'S_1','*Movie*') ∧
HasTitle#2[*Movie#1,String*](*y*,*title*))

where we have assumed that 'S_1', the name of the data source, is the name of the agency and '*Movie*' the name of the identification scheme. Both are constants and, of course, they can be changed.

An indirect attribute A such that

map(A) = $(P_1(E_1,O),P(O,T))$

is represented in ϕ by the formula

$\exists\,!z\;(O(z) \wedge P_1[E_1,O](y,z) \wedge P[O,T](z,t))$

As an example, consider R_8 in which we have

map(*director*) =
(*HasDirector#4*(*Movie#1*,*Director#4*),
HasName#1(*Director#4*,*String*))

where *HasName#1* is a simple global identifier of *Director#4*. The source description of R_8 is then:

MovieDirectors(*title,dir*) →
∃ !*y*(*Movie#1*(*y*) ∧

$HasTitle\#2[Movie\#1,String](y,title) \wedge$
$\exists\,!z(Director\#4(z) \wedge$
$HasDirector\#4[Movie\#1,Director\#4](y,z) \wedge$
$HasName\#1[Director\#4,String](z,dir)))$

When an indirect attribute maps to an entity type that has a composite identifier, then all attributes in R of that identifier are represented together in ϕ after the initial part:

$\exists\,!z\,(O(z) \wedge P_1[E_1,O](y,z) \wedge \ldots$

5.3 Assessment

We have shown that the mediated schema and the source descriptions can be automatically generated from the mappings of the relational schemas to the UO. The main drawback we see is that some predicate names of the mediated schema may be not user-friendly, which is a problem if there are human users. In this case, the predicate names will need to be manually improved by the designers.

6 Related work

In this section we review two precursor systems that used a kind of universal ontology for data integration: Carnot and SIMS.

As far as we know, Carnot (Collet et al. 1991; Huhns et al. 1993) was the first system that maps local schemas to a pre-existent global schema, which in this case it is the Cyc ontology. Each local schema is mapped to Cyc, independently of other local schemas. Carnot includes a tool (Model Integration Software Tool) that assists users in finding the correspondences of each concept of the local schema with a Cyc concept. Using those correspondences, the tool automatically generates articulation axioms, which formally state the mapping between the instances of the local source and Cyc's knowledge base.

In Carnot there are not mediated schemas. The local schemas are integrated into Cyc, extending it if needed. The consequence is that the data content of integrated system is not explicit, and therefore users may find difficult to query it (Collet et al. 1991, p. 62).

The work most closely related to ours is SIMS (Arens et al. 1993; Arens et al. 1996). SIMS uses two kinds of models, both written in the Loom language. The first is a domain model, which describes the classes in the domain and their relationships. The second is the data source model, which describe the classes and relationships in each data source.

The mapping between a data source model and the domain model is done by defining a data source link, IS-link, between the concepts and roles in both models. The meaning of an IS-link between a data source class and a domain class is that the two classes contain exactly the same set of individuals, although the data source class might contain only a subset of the attributes for the class. The links between the roles indicate that those roles have the same meaning for the linked classes.

In SIMS, the minimal model of a data source is a model that includes a data source model and enough of a domain-level model to exactly cover the data source model. The mediated schema, called minimal model, is then the union of all minimal models for all the data sources available to the system. Informally, the minimal model is the smallest model that can describe the semantics of, and provide access to, the entire contents of the available data sources.

SIMS shows that, when a domain model is available and the data sources are mapped to it, the design of the mediated schema and the source descriptions is easier. Our approach is similar to that of SIMS, but we do not need to model the data sources and the LAV mappings of the data sources to the universal ontology can be more expressive.

7 Conclusions

We have tried to show that the universal ontology (UO) solves, or at least greatly alleviates, the semantic integration problem. To this end, we have presented an approach to data integration based on the UO.

Our approach requires that the developers of the data sources define the mappings between the data source schemas and the UO. We have argued that this is easy if the concepts are in the core part of the UO, although it may be not so easy if they must be composed. However, this must be done only once per relational schema, and it is useful as a documentation of the semantics of the schema. On the other hand, some kind of mapping is needed in all approaches.

We have shown that from these mappings, the other semantic components of a data integration system (mediated schema, source descriptions) can in principle be automatically generated. This is the main advantage of the UO-based approach, which confirms that indeed the UO solves, or at least greatly alleviates, the semantic integration problem.

There are some aspects of the semantic integration problem that we have not discussed in this paper, such as integrity constraints, local values, local completeness or query expressions, but we believe that they would not significantly change the overall conclusion.

References

An Y., Borgida A., Mylopoulos J. (2006) Discovering the Semantics of Relational Tables Through Mappings In: Journal on Data Semantics VII Spaccapietra S. (ed.) Springer Berlin Heidelberg, pp. 1–32 https://doi.org/10.1007/11890591_1

Arens Y., Chee C. Y., Hsu C.-N., Knoblock C. A. (1993) Retrieving and Integrating Data from Multiple Information Sources. In: International Journal of Cooperative Information Systems 02(02), pp. 127–158

Arens Y., Knoblock C. A., Shen W.-M. (1996) Query reformulation for dynamic information integration. In: Journal of Intelligent Information Systems 6(2), pp. 99–130

Batini C., Ceri S., Navathe S. B. (1992) Conceptual Database Design: An Entity-relationship Approach. Benjamin-Cummings Publishing Co., Inc.

Collet C., Huhns M. N., Shen W. M. (1991) Resource integration using a large knowledge base in Carnot. In: Computer 24(12), pp. 55–62

Doan A., Halevy A., Ives Z. (2012) Principles of Data Integration. Morgan Kaufmann, Burlington

Fowler M. (1997) Analysis Patterns: Reusable Object Models. Addison-Wesley

Grüninger M., Uschold M. (2004) Ontologies and Semantics for Seamless Connectivity. In: SIGMOD Record 33, pp. 58–64

Huhns M. N., Jacobs N., Ksiezyk T., Shen W.-M., Singh M. P., Cannata P. E. (1993) Integrating enterprise information models in Carnot. In: Proceedings International Conference on Intelligent and Cooperative Information Systems, pp. 32–42

Lenzerini M. (2002) Data Integration: A Theoretical Perspective. In: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems,PODS ’02. ACM, pp. 233–246 http://doi.acm.org/10.1145/543613.543644

Lockemann P., Mayr H., Weil W., Wohlleber W. (1979) Data Abstractions for Database Systems. In: ACM Transactions Database Systems 4, pp. 60–75

Madnick S. E. (1996) Are we moving toward an information superhighway or a Tower of Babel? The challenge of large-scale semantic heterogeneity. In: Proceedings of the Twelfth International Conference on Data Engineering, pp. 2–8

Mena E., Illarramendi A., Kashyap V., Sheth A. P. (2000) OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies. In: Distributed and Parallel Databases 8(2), pp. 223–271 http://doi.acm.org/10.1023/A:1008741824956

Michael J., Mayr H. C. (2017) Intuitive Understanding of a Modeling Language. In: Proceedings of the Australasian Computer Science Week Multiconference, ACSW ’17. ACM, Geelong, Australia, 35:1–35:10 http://doi.acm.org/10.1145/3014812.3014849

Miller G. A. (1995) WordNet: A Lexical Database for English. In: Commun. ACM 38(11), pp. 39–41 http://doi.acm.org/10.1145/219717.219748

Olivé A. (2007) Conceptual modeling of information systems. Springer-Verlag Berlin Heidelberg http://doi.acm.org/10.1007/978-3-540-39390-0

Olivé A. (2017) The Universal Ontology: A Vision for Conceptual Modeling and the Semantic Web (Invited Paper) In: Conceptual Modeling: 36th International Conference (ER 2017) Mayr H. C., Guizzardi G., Ma H., Pastor O. (eds.) Springer International Publishing, pp. 1–17 https://doi.org/10.1007/978-3-319-69904-2_1

Park J., Ram S. (2004) Information Systems Interoperability: What Lies Beneath? In: ACM Transactions on Information Systems 22(4), pp. 595–632 http://doi.acm.org/10.1145/1028099.1028103

SAP (2016) Identifiers. https://help.sap.com/saphelp%5C_ewm94/helpdata/en/48/d0060005ae154ee10000000a421937/frameset.htm

UN/CEFACT (2009) Core Components Data Type Catalogue Version 3.0.. https://www.unece.org/fileadmin/DAM/cefact/codesfortrade/CCTS/CCTS-DataTypeCatalogueVersion3p0.pdf

Uschold M. (2000) Creating, Integrating and Maintaining Local and Global Ontologies. In: Proceedings of the First Workshop on Ontology Learning (OL-2000) in conjunction with the 14th European Conference on Artificial Intelligence (ECAI-2000)