J Intell Inf Syst (2014) 43:437–462
DOI 10.1007/s10844-014-0340-5

Data-centric intelligent information integration—from concepts to automation

Matthias Jarke · Manfred Jeusfeld · Christoph Quix

Received: 21 December 2012 / Revised: 9 January 2014 / Accepted: 14 October 2014 /
Published online: 29 October 2014

Abstract Intelligent integration of information continues to challenge database research for over 35 years. While data integration processes of all kinds are now reasonably well understood and widely used in practice, the growth and heterogeneity of data requires much higher degrees of automation to limit the need for human specialist work. This requires deeper insights in data-centric approaches of Enterprise Information Integration which focus on the semantics of information integration. Recent formalizations and algorithms enable both significant improvement in schema integration, and in its automated transformation to efficient data-level integration, in a wide variety of architectural settings such as data warehouses or peer-to-peer databases. In addition to giving a short overview of developments in this field for the past 20 years, this paper focuses particularly on the challenges posed by heterogeneity in data models.

Keywords Metadata · Data integration · Role-based model · Model management

1 Introduction

Even after 35 years of database research, information integration remains one of its key challenges Shu et al. 1977. Traditionally, there have been two different foci in this research. A focus on the *process of data integration*, for example the often cited ETL (extract-transform-load) processes in data warehousing, is nowadays a well-established industry sector called *Enterprise Application Integration* (*EAI*). Among other factors, it was driven by resource constraints that often made the scheduling of integration tasks (in order to optimally exploit the buffer space available for intermediate integration results) the most critical bottleneck in information integration.

M. Jarke
RWTH Aachen University, Informatik 5, Ahornstr. 55, 52074 Aachen, Germany

M. Jarke (✉) · C. Quix
Fraunhofer FIT, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
e-mail: jarke@informatik.rwth-aachen.de

M. Jeusfeld
University of Skövde, School of Informatics, 54128 Skövde, Sweden

In contrast, the *data-centric approach* to information integration focuses on the semantics of the integrated information, in practitioner terms: the *data quality*. Classical techniques include the definition and analysis of formal constraints across data sources, in order to identify semantic conflicts or overlaps that could be exploited for data cleaning and integration. In industry, this strain—in the Internet propagated under the label of the Semantic Web—has caught on much later than EAI but is nowadays gaining importance under the label of *Enterprise Information Integration* (*EII*). One important motivation is obviously the enormous costs of dealing with data quality problems. Even more importantly, advances in data mining algorithms still need to rely on the exploitation of high-quality historical and sensor data for successful pattern recognition and prediction in science and industry.

For decades, most integration tasks were solved manually, with rather limited formal or tool support. Recent practice surveys have claimed that about 40 % of database-related work in industry is spent on data integration issues Brodie 2010, an issue so important that it caused top management attention with 68 % of CEOs surveyed by IBM Haas 2007.

The increasing complexity in terms of data volume, heterogeneity, and especially size and number of models, poses new challenges to the design and the development of integrated information systems. Fortune-500 enterprises now employ several thousand database systems with a few hundred relations each Brodie 2010. In such settings, manual approaches to information integration are no longer feasible Haas 2007; Smith 2007; Bernstein and Haas 2008. However, especially more automation in Enterprise Information Integration requires a deeper formalization of the semantic foundations e.g. in logic Lenzerini 2002. Even informal tasks such as the identification of shared or similar elements in different schemas must be formalized somehow, albeit with many different techniques, leading to the development of a complete new subarea of research and industry called *schema matching* or *ontology matching* Rahm and Bernstein 2001.

In addition, while most database systems still use the relational data model, data sources and applications may include a broad range of data formats, informal media objects, and a wide variety of different modeling and metadata languages both in their operation and in their design and evolution. The heterogeneity problem becomes even more intense in mobile multimedia applications with data as well as service integration requirements.

Since the turn of the century, the research area of *model management* Bernstein et al. 2000 therefore aims at high-level methods and automated systems to support the development of metadata-intensive applications. A typical example is the definition of a *model algebra* that provides high-level abstract operators for the key model-level tasks underlying data integration:

- *Match*: the identification of correspondences between models (match, Rahm and Bernstein 2001; Shvaiko and Euzenat 2005),
- *Compose*: the (possibly multi-step) transformation between models based on specifications of their inter-relationships as a formal mapping Miller et al. 2000; Arenas et al. 2010,
- *Merge*: the integration of models (schema merge, Batini et al. 1986; Parent and Spaccapietra 1998), and
- the actual *execution* of the specified data transformations Melnik et al. 2005; Haas et al. 2005.

In this paper, we present a brief review of the evolution from classical data-centric integration to the recent advances in integrated model and data management enabled by more than 20 years of research in intelligent information integration. In Section 2, we summarize classical data integration efforts up to the data warehouse movement of the late 1990's. In

Section 3, the evolution of model management from precursors in the mid-1990s until today is reflected with a discussion of some of the most important ideas and prototypes. At the boundary between both phases, we briefly review the history of our ConceptBase system for which a key paper appeared in JIIS 1995 Jarke et al. 1995, and became one of the most-cited papers in the journal's history.

Section 4 focuses on the challenge of automatically dealing with heterogeneity in model management. A formal metamodeling framework and model management toolset must simultaneously support the model management operators and their automated data-level executability for a broad range of modeling and operational data languages under which the systems-to-be-integrated might be operated or (re-)designed. As an example, we describe our Generic Role-Based Metamodeling suite and its underlying theory. In the final Section 5, we present the application of this approach to classical data integration problems such as schema matching and schema merging.

As a running example, we choose a case study in mobile traffic data integration Geisler et al. 2012 shown in Fig. 1. In this scenario, data streams from mobile devices or sensor networks have to be integrated with data from classical database systems and web services. For example, consider a traffic information system which makes use of various information sources to derive accurate information of a current traffic situation. In case of an accident or a traffic jam, cars send messages (Floating Car Data FCD[1] Kerner et al. 2005) of the event or their current state to a traffic information system which integrates, aggregates, and analyzes the received messages in real time in the context of existing database information.

Such context data might come from heterogeneous external information sources, e.g., weather information to check the consistency of temperature data delivered by sensors in roads, road side units, or cars; traffic density information derived from aggregated C2X messages is complemented by data from TMC (Traffic Message Channel) or a database of road construction sites; a baseline for the traffic state can be derived from historical information.

2 Classical data integration

The classical procedural approach to data integration implicitly assumes a staged software architecture which was made explicit in the early 1990s as the so-called mediator architecture as depicted in Fig. 2 Wiederhold 1992. Data from several sources is integrated by a mediator which might follow a virtual or materialized integration approach.

In a *virtual integration* scenario first mentioned in the distributed database context by Ceri and Pelagatti 1984, the mediator must reformulate the queries of the applications and integrate the data from sources on the fly. The application queries are expressed in terms of the global schema of the mediator. To retrieve the data from the source, they have to be translated into queries in terms of the local schemas of the sources. Wrappers take these reformulated queries, send them to the sources, extract the answers, and send the result back to the mediator. Wrappers may also apply some simple transformations such as translating the answers into a uniform format.

In the *materialized integration* scenario Zhou et al. 1996, the data is stored in a central data repository, since the early 1990s called a data warehouse Jarke et al. 2003. While research has

[1] FCD is transmitted using some Car-To-Car (C2C) or Car-to-Infrastructure communication service (C2I, Stubing et al. 2010). C2C and C2I communication is summarized under the term C2X (Car-to-X) communication.

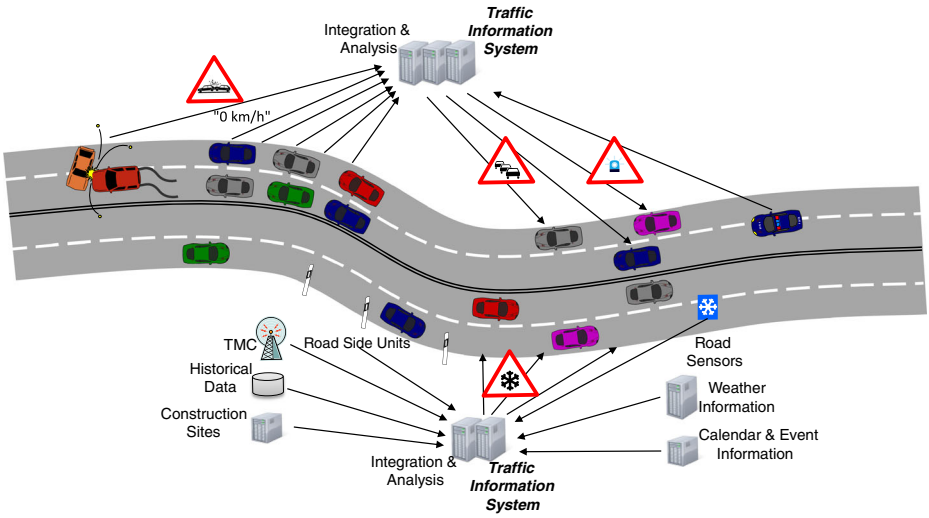

Fig. 1 Information integration in a traffic information system using C2X communication

focused on the fundamental and transformational aspects of this, very similar to the virtual integration scenario, industrial practice is at least equally interested in the resource-constrained scheduling of the huge bulk tasks involved in operating this architecture with the enormous data sizes of today. So-called ETL tools support the main steps within such a process which actually predated data warehousing by over 10 years (EXPRESS Shu et al. 1977): *Extract* source data into some buffer, *Transform* them by cleaning, model unification, and merging (mediator), and *Load* them to the data warehouse.

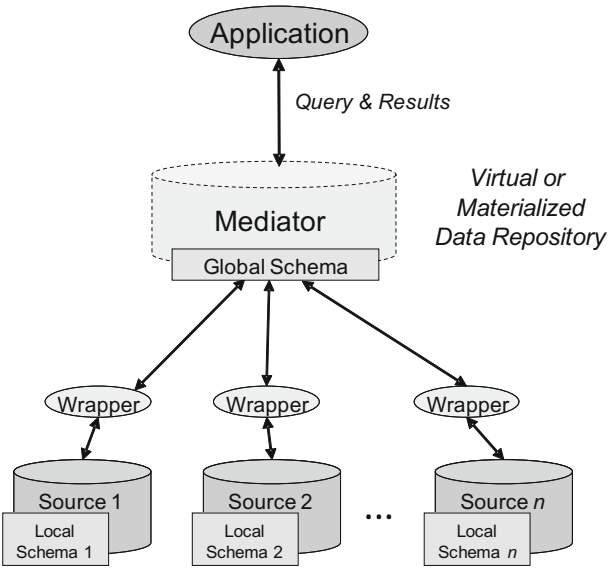

Fig. 2 Architecture of an information integration system

On the formal side, integration methods in the 1990s began to study explicit formal mappings between the source and integrated data; for an early overview, see Wiederhold 1996. Such mappings could serve as a basis for "model-based" data integration where wrapper and mediator code could be generated largely automatically from the mapping specifications (e.g., Carnot Collet et al. 1991; Singh et al. 1997, Infomaster Genesereth et al. 1997, Information Manifold Kirk et al. 1995, SIMS Arens et al. 1996). While Infomaster and the Information Manifold focused on the Relational Data Model as modeling language, Carnot and SIMS used description logics Nardi and Brachman 2003 to express relationships on the model level as well as on the data level.

There are many ways to characterize the mappings between schemas. The simplest form is correspondences between individual attributes of the schemas. This type of mapping is frequently used in schema matching Rahm and Bernstein 2001 but cannot be used directly for information integration because it does not capture more complex relationships such as restructuring and regrouping of data. Since the logic-based data warehouse research of the late 1990s, mappings are therefore often expressed as a set of query pairs. In each pair, a query q_S over a source schema S is related to a query q_G over the global schema G: $q_S \sim q_G$ Lenzerini 2002. The relationship "~" between the queries is a set relationship like = or ⊆, which means that the result set of the query q_S is equivalent to (a subset of, resp.) of the result set of query q_G for all valid database instances.

There are two semantic perspectives for formalizing the mappings, called *Local-As-View* (*LAV*) and *Global-As-View* (*GAV*). In GAV mapping, any single element g of the global schema is defined as a view on the sources, i.e., $q_S \sim g$. This reflects the classical data integration perspective of the mediator approach. In the LAV approach, an element s of the source schema is defined as a view on the global schema, i.e., $s \sim q_G$. This reflects the idea of the integrated data as a partial description of a uniform "real world" about which the data sources capture perhaps incomplete, erroneous or even inconsistent observations; LAV therefore seems more relevant when we talk about important real-world problems such as semantic data quality.

At first glance, query rewriting in the case of GAV seems to be easier as a query over the global schema needs just to be unfolded, i.e., the elements of the global schema in the query are replaced with the corresponding query q_S over the sources Lenzerini 2002. However, in the case of constraints and incomplete sources (which is a common assumption in information integration) more complex reasoning is required to answer queries Calì et al. 2004.

In LAV, query rewriting corresponds to the problem of answering queries using views Halevy 2001, which also requires reasoning; however, starting with MiniCon Pottinger and Halevy 2001, a number of efficient algorithms have been developed. Some integration architectures which go beyond the mediator schema of Fig. 2, such as Peer-to-Peer data management systems Halevy et al. 2004, require a combination of GAV and LAV mappings called GLAV.

In parallel to these developments, the growth of the Internet caused a rather different approach to intelligent information integration to emerge. The lack of central planning and the resulting irregularity of data structures in web-based systems require more flexible approaches for data management. TSIMMIS was one of the first projects that moved away from formal schemas and schema mappings as the basis for integration. Instead, it used self-describing semi-structured graph data models Garcia-Molina et al. 1997, later replaced by the slightly more restrictive, tree-oriented XML standard. Similar to the approach taken by modern search engines, data integration in such a setting became a mix of text retrieval and graph mappings without higher-level schemata. Even though this kind of approach persists until today (as matching algorithms, see below in Section 5.1), it cannot easily exploit the rich

knowledge available in the schemas of the ten thousands of databases and ontologies available today, or the documented design models which give even more semantics to these schemas. To address these challenges, model management emerged at the turn of the century.

3 The evolution of model management

The creation of models and mappings in the classical data integration systems was largely a manual task. While automated support had been proposed for a few integration tasks (e.g., schema integration Batini et al. 1986 or schema matching Rahm and Bernstein 2001), the design, implementation, and maintenance of the integration system had to be done manually. This might have been acceptable for data management systems with a manageable schema complexity, but the systems have grown significantly in the recent years. The complexity of current "information eco-systems" Brodie 2010 with heterogeneity at various levels requires a methodical support for the management of models and mappings.

The importance of data models in the development of integrated information systems has been recognized by Bernstein et al. 2000 in their vision of *model management systems*. In such systems, models should be regarded as first-class objects, and the systems should provide operators to "work" with these models. For example, a *Match* operator should be used to compute a mapping between two models, the *Merge* operator should integrate models based on a given mapping, and the *ModelGen* operator should generate a new model by translating a given model into another modeling language. The vision was an algebra which would allow an abstract specification of complex operations on data models.

3.1 Model management 0.1: repositories with formal metamodels

The vision of model management in Bernstein et al. 2000 initiated new research in this area, but there has been significant research on individual topics before Quix et al. 2009.

The earliest approaches to schema integration and matching are summarized in Batini et al. 1986. They were mainly based on abstract, conceptual modeling languages (e.g., variants of the Entity-Relationship model), or directly operated on the relational schemas with intra-relational and inter-relational dependencies Casanova and Vidal 1983; Biskup and Convent 1986. In both cases, the mapping languages were rather weak as only one-to-one correspondences could be expressed.

With the increasing size and complexity of database systems in the late 1980s, the necessity for formal methods for the management of complex data models became evident. Business IT researchers like Dolk 1988 first stated the requirement for a theory for models similar to the relational database theory. Such a theory should include formal definitions of models and operations on models and could be used as a basis for the implementation of a model management system. This work was based on a draft of the *Information Resource Dictionary System* (ISO/IEC 1990) standard ISO/IEC 1990 which was accepted by ISO in 1990. IRDS clarified the terminology of modeling systems as a four-level hierarchy. A decade later, the Unified Modeling Language (UML) community adopted the same approach with slightly different terminology in its MOF standard (Meta Object Facility ISO/IEC 2005). The metamodel hierarchy in Fig. 3 shows a MOF-based repository hierarchy for our running example: at the lowest level reside data instances which are described by a model (or schema) on the next higher level. The model is expressed in some modeling language (or metamodel) which is located at the third level. The highest level contains a metametamodel which can be used to define metamodels.

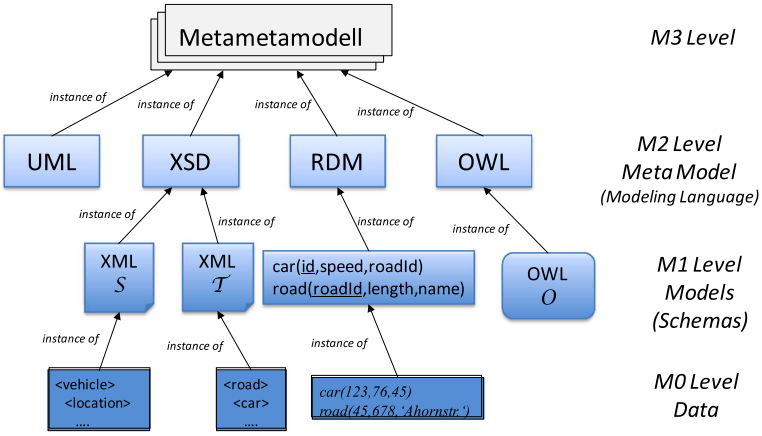

Fig. 3 Metamodeling hierarchy according to the meta object facility ISO/IEC 2005

The IRDS framework introduced the concept of so-called metadata repositories for purposes such as development process traceability, information integration, and model transformation Quix 2009a. For example, solutions for forward and reverse engineering between ER models and relational databases were developed using generic metametamodels Atzeni and Torlone 1996; Jeusfeld and Johnen 1995, i.e., a uniform representation at the M3 level). Generic metamodels at the M2 level also addressed several aspects of database schema integration Spaccapietra and Parent 1994; Pottinger and Bernstein 2003. However, these early solutions mainly concentrated on the transformations at the model level and did not pay detailed attention to automated transformation on the data level.

In parallel to the initial IRDS standardization, a logic-based approach to dealing with an unbounded number of meta levels was investigated in the Telos project jointly conducted between the University of Toronto and several European projects in the late 1980s Mylopoulos et al. 1990. An important feature of Telos is the strong and highly efficient formalization in Datalog with stratified negation Jeusfeld 1992 from which excerpts are briefly reviewed in Section 4.3. Based on this formalization, we developed the deductive metadatabase system ConceptBase whose complete description was published in JIIS 1995 Jarke et al. 1995 and whose present version is still widely used in several thousand installations worldwide Jarke et al. 2009. Based on the Datalog formalization, ConceptBase was the first metadata repository to also offer effective query optimization, integrity constraint evaluation, and incremental view maintenance at the data level Staudt and Jarke 2000, as well as viewpoint resolution Nissen and Jarke 1999 and requirements traceability Ramesh and Jarke 2001 at the meta level simultaneously, thus providing an early example of fully automated model-based code generation.

With the confluence of structured data, text and multimedia capabilities, the World Wide Web, and mobile communications, the range of data models has grown well beyond what could be covered by these early approaches. In Haslhofer and Klas 2010, an overview of metadata interoperability in heterogeneous media repositories is given. Although the survey focusses on media repositories, it also addresses interoperability between "structural" modeling languages, such as UML, XML Schema, and OWL. The authors classify approaches according to the MOF hierarchy and argue that effective interoperability between systems can only be achieved if data transformations at the instance level are also addressed. Furthermore, they distinguish between standardization and mapping approaches. The former propose

metadata standards to enable interoperability, whereas the latter build relationships between different metamodels. Mapping approaches are more complex, but are advantageous in open environments such as the Web, as in these cases, no central authority can enforce a standard Haslhofer and Klas 2010.

3.2 Model management 1.0: algebra of model operators

The increasing complexity of information systems requires techniques for automating the tasks of creating models and mappings. The original vision of model management aimed at providing support for these tasks Bernstein et al. 2000, even though a complete automation was expected to be hard to achieve. The creation of models and mappings was considered a design activity which requires a deep understanding of the semantics of the modeled systems. Such tasks are considered "AI-complete", i.e., it requires human intelligence to solve these problems Bernstein et al. 2004. Another important motivation for the definition of a model management algebra was the observation that many applications that deal with models require a significant amount of code for loading and navigating models in graph-like structures. A model management system based on a formal algebra should simplify the development of model-oriented applications in the same way as data management systems based on relational algebra simplified the development of data-oriented applications Bernstein et al. 2000.

First model management systems such as Rondo Melnik et al. 2003a; b and COMA Do and Rahm 2002 applied simple, abstract model representations in which a model is represented as a directed, labeled graph. Other approaches for model management include a large body of research on schema matching Rahm and Bernstein 2001; Shvaiko and Euzenat 2005. Although a graph representation is often sufficient for basic schema matching tasks, semantic details of the models (such as constraints) cannot be easily represented. Mappings are often just represented as a set of pairwise correspondences between nodes in the graphs. MISM (Model-Independent Schema Management) uses a richer representation for schemas Atzeni et al. 2009. Schemas are described in a generic way using a multi-level dictionary. However, the system uses a set-theoretic approach for some model management operators (e.g., *Merge*), i.e. again only correspondences ('equivalence views' in the terminology of MISM) are used as the mapping formalism.

In the original vision of model management Bernstein et al. 2000, mappings had a weak representation and were seen as a special type of a model which might include expressions to describe the semantics of a mapping in more detail (e.g., by using a SQL query). However, to automate operations on models and mappings, mappings have to be represented in a separate formalism which is more expressive than just simple correspondences.

3.3 Model management 2.0: mappings as first-class citizens

There have been several attempts aiming at combining a rich modeling language with powerful mapping languages. For example, in the European DWQ project (Foundations of Data Warehouse Quality Jarke et al. 2003), a semantically rich metamodel Jarke et al. 1999 was combined with an information integration approach based on description logics Calvanese et al. 2001. Similarly, the Italian MOMIS system used an object-oriented modeling language to support the integrated querying of heterogeneous information sources Bergamaschi et al. 2001.

The Clio project between IBM and the University of Toronto Hernández et al. 2001; Haas et al. 2005; Fagin et al. 2009 introduced a strong mapping language based on tuple-generating dependencies (tgds) Beeri and Vardi 1984; Abiteboul et al. 1995. The well-defined, formal basis and the ability to easily translate the mappings into executable code (e.g., queries in SQL

or XQuery) proved a significant advantage and caused a re-thinking of the whole definition of model management, dubbed Model Management 2.0 Bernstein and Melnik 2007.

In model management 2.0, it has been realized that the *representation of mappings* is at least equally important as the representation of models Bernstein and Melnik 2007. Mappings are involved in all model management operations, and mappings are at the core of any integration approach. Furthermore, model management is not only a design time issue. The runtime system has also to be taken into account, because mappings have to be executed eventually to perform data transformation tasks. Thus, it is not sufficient to hide the semantics of a mapping in a string expression in some arbitrary language. Model management systems must be able to understand the semantics of a mapping in order to enable mapping operations (e.g., composition, inversion) and produce mappings as output (e.g., in match and merge operations). While Clio mostly explored this issue in the context of (nested) relational data models, the following section will investigate the extension to the management of heterogeneous data models.

4 Towards heterogeneous model management

The heterogeneity of data management systems and modeling languages used in the web, but also in enterprise information systems, requires a mapping language, which is able to cope with the different modeling formalisms in a single uniform framework. Moreover, this has to be done at the same time at the model level as well as the data level.

A heterogeneous model mapping states how the data of one model is related to the data of another model. It is important to note that mappings relate data and not only models. Because of this, mappings need to be very expressive in order to be able to represent rich data transformations. *Executability* of a mapping language means that it must be possible to apply a mapping such that it enables automatic code generation that executes the data transformations specified in the mapping.

Summarizing, a model management system needs to address at least the three lower levels of Fig. 3: the data level for expressive data translations, the model level for operations on models such as *Match* and *Merge*, and the metamodel level to enable a generic representation of heterogeneous data models.

Defining a mapping language between all individual pairs of different modeling languages would be a daunting if not impossible task, as for each pair, syntax and semantics of two individual formalisms have to be interlinked. A generic modeling language simplifies this task by providing a uniform basis for the definition of mappings. Model management operations have to be based on formal languages with rich semantics. These formalisms are necessary in order to support the development of model management systems which have to produce in the end mappings and models in existing, formal languages (e.g., SQL, XML Schema, XQuery). Moreover, a formal basis is required to prove characteristics of model management operations, e.g., that a model transformation is correct and complete, or a merged schema is minimal, but also preserves all information of the input schemas.

Last not least, there is a trend that the restrictions of relational and even XML database systems are too tight for new applications that require high availability and scalability for the web Vogels 2007; Stonebraker 2010. Thus, new data management systems with another set of modeling languages are being introduced (e.g., not-only-SQL or NoSQL database systems Cattell 2010), which again require transformations and mappings of existing data models. Consequently, the management of heterogeneous data models will be a running challenge also for future information integration projects, and model management systems must be extensible and flexible to support also future modeling languages.

To illustrate the concepts and algorithms in the following subsections, Fig. 4 extracts a simplified heterogeneous schema integration task from the scenario in Fig. 1. An XML web service *S* provides information about vehicles and their location. A relational data stream *R* gives Floating car information about cars, their speed and the roads on which the cars are traveling. These two sources should be integrated into an XML database *T*, which has information about roads and for each road a list of cars which are driving on this road. The XML documents use a different vocabulary, therefore, we might use an ontology *O* as the semantic bridge between the XML documents. The relational schema *R* is mapped directly to the target XML schema *T*.

4.1 Requirements for a generic model management system

A generic representation of models is a prerequisite for building a model management system. Without a generic representation, model operations would have to be implemented for each modeling language that should be supported by the system. Especially for the task of model transformation, a generic representation of models is advantageous as the necessary transformations have just to be implemented for the generic representation. Such a generic representation is called a *generic metamodel*. A generic metamodel should be able to represent models originally represented in different metamodels (or modeling languages) in a generic way without losing much detailed information about the semantics of the model.

First implementations of model management systems used rather simple graph representations of models, e.g., Rondo Melnik et al. 2003b. Although the graph-based approach might allow an efficient implementation of operations which do not rely on a detailed representation of the models (such as schema matching), it makes it more difficult to implement more complex operations (such as model transformation or schema integration). Schema integration methods often use rather abstract metamodels extending traditional conceptual modeling formalisms, such as the ERC + model in Spaccapietra and Parent 1994. They usually focus on the conflicts at the semantic level, because such modeling languages are good at representing the semantics of a model. Schema integration approaches using a more concrete metamodel (e.g., the relational model with constraints and dependencies Casanova and Vidal 1983; Biskup and Convent 1986) assume that these semantic conflicts have already been resolved and provide solutions for integrating schemas without conflicting constraints.

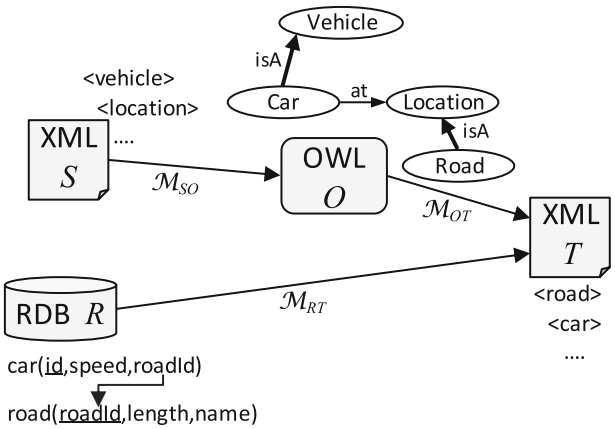

Fig. 4 A simple, heterogeneous data integration scenario

It is apparent, that a richer modeling language is required for model management operations that explicitly deal with the semantics of model elements in a heterogeneous setting. Early examples of such metamodels have been used, e.g., for model transformation Atzeni and Torlone 1996; Atzeni et al. 2006; Mork et al. 2007.

In Atzeni and Torlone 1996, the authors describe a metamodel consisting of "superclasses" of the modeling constructs in the native metamodels. The transition between this internal representation and a native metamodel is described as a set of patterns. This induces the concept of a *supermodel* which is the union of patterns defined for any supported native metamodel. This model representation has been expressed in a relational model dictionary Atzeni et al. 2005, and was used for the generic *ModelGen* implementation *MIDST* Atzeni et al. 2006.

In Jeusfeld and Johnen 1995, a ConceptBase metametamodel to enable the translation of models between different modeling languages is used. A model element in a concrete modeling language is mapped to the metametamodel, in which models can be rearranged, and then translated into the desired target modeling language. Another metamodel following the approach of generalizing metaclasses is Vanilla Pottinger and Bernstein 2003 which has been used to implement model merging.

4.2 Role-based metamodeling with *GeRoMe*

Our metamodel *GeRoMe* provides a generic, yet detailed representation of data models originally represented in different languages Kensche et al. 2007a. In its role-based modeling approach Bachman and Daya 1977; Richardson and Schwarz 1991; Wong et al. 1997, an object is regarded as playing roles in collaborations with other objects. This allows describing the properties of model elements as accurately as possible while using only metaclasses and roles from a relatively small set. This strongly reduces a well-known problem in metamodeling, as follows.

A classical approach for modeling a generic metamodel could define a hierarchy of metaclasses that represent an abstraction of concrete modeling elements in existing modeling languages (e.g. Jeusfeld and Johnen 1995. One could then map the concrete modeling elements to exactly one of these metaclasses. However, modeling elements in different modeling languages have often similar or overlapping semantics, but rarely a truly equivalent semantics. When two model elements from different metamodels are mapped to the same metaclass in the generic metamodel, this implies that the elements have the same semantics in the view of the generic metamodel. Thus, differences and details of model elements are lost due to the abstraction in the generic metamodel. A solution to this problem could be to model the detailed semantics of model elements by intersection classes, i.e., classes which inherit features from several base classes representing basic modeling features (e.g., aggregation, inheritance, association). However, many intersection classes would be necessary to represent all the different combinations of modeling features which are present in one concrete modeling element. In earlier works, e.g., in our interdisciplinary research on conceptual modeling of chemical engineering processes Baumeister and Jarke 1999; Brandt et al. 2008, this has been shown to lead, even in practice, to a combinatorial explosion of subclasses.

GeRoMe's role-based approach represents each modeling feature by a separate role class. Model elements are plain objects which do not have any semantics by their own. By decorating a model element with role objects, a model element gains the modeling features of these role objects. This allows an arbitrary combination of modeling features.

The implementation of model management operators is simplified as it can focus on the roles which are relevant for a specific operator. Roles provide a view, i.e., a subset of the

features of a model element. For example, when elements should be matched by name, the match operator needs to consider only the role providing the name of the element, all other roles can be ignored. Another advantage of the role based modeling approach is that roles—and thereby modeling features—can be easily added to or removed from a model element without changing its identity. This characteristic is in particular important for model transformation.

Consider again the example in Fig. 4. Simplified *GeRoMe* representations for the relational model *R* and the XML Schema *T* are shown in Fig. 5. Gray boxes denote model elements, white boxes attached to them are role objects. The left part of the figure illustrates the *GeRoMe* model for the relational schema. The light gray elements car and road represent the model elements for the two relations *car* and *road*. Both elements play the role Aggregate (abbreviated by Ag), meaning that they can have attributes. In addition, they play the role Referable (R), which allows them to be the target of a key constraint. The attributes are represented by the medium gray model elements; all of them play the role Attribute (Att). In addition, roadId in car plays the role Reference (Ref) as it is a foreign key attribute. For simplicity, we show only the domain for the roadId attributes; it is the model element Integer which plays a Domain role (D).

Constraints are represented by dark gray elements. The primary key constraints PKcar and PKroad play the roles Injective (Inj) and Identifier (Id) as they represent uniqueness constraints which, in addition, identify the referenced type. Furthermore, the foreign key FKroad points to the reference role of the corresponding attribute.

The representation of the XML schema is more complex, as it contains more structural information. Complex types in XML-Schema are similar to classes in UML or entity types in the ER model, as they can have attributes and participate in associations. Elements in XML represent associations, i.e. a relationship between two complex types (a nested and a nesting

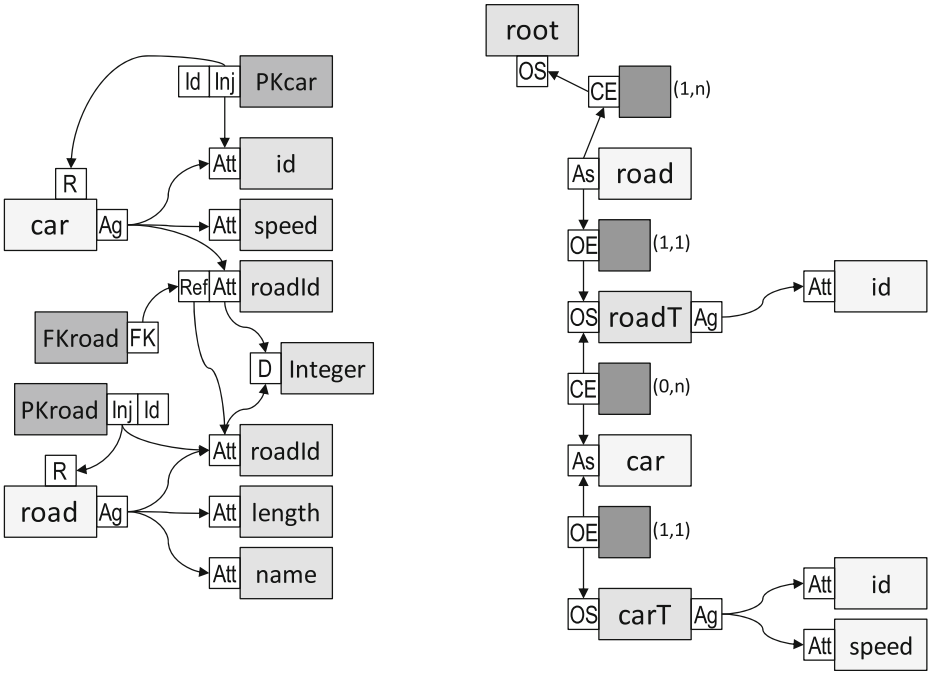

Fig. 5 Simplified *GeRoMe* models for the relational and XML schema from Fig. 4

type). Because of XML's tree structure, an instance of a complex type is always nested into exactly one parent element. In the right part of Fig. 5, the complex types roadT and carT also play Aggregate roles as the corresponding constructs in the relational model. In addition, they play ObjectSet (OS) roles as they also may participate in associations. In the example, road and car play the Association role (AS) and represent the XML elements. They link the complex types (and the document root) using association end roles, as indicated by unlabeled dark gray boxes. There are two different types of association ends in this example: CompositionEnds (CE) state the relationship to the parent element and ObjectAssociationEnds (OE) represent the link to the child element. These elements also maintain the information about the cardinality constraints of this association. Associations in UML or relationship types in ER are modeled in *GeRoMe* using the same set of roles.

4.3 Logical foundation for the generic metamodel

Based on our experience with Telos Mylopoulos et al. 1990 and ConceptBase Jarke et al. 1995; Jarke et al. 2009, the formalization of *GeRoMe* represents a model as a set of logical facts. Each fact defines a model element, a role, a property, or a relationship between role objects. This logical representation allows declarative specifications and sound and efficient implementation of some model management operators.

A generic metamodel must provide a transformation between the concrete modeling languages and its generic representation. *GeRoMe* enables this import and export of models in a declarative way: equivalence rules state that a combination of modeling constructs in a particular modeling language are equivalent to a set of model elements and role objects in *GeRoMe* Kensche and Quix 2007. As in Jeusfeld 1992, the translation between *GeRoMe* and specific modeling languages is based on Datalog rules Kensche and Quix 2007. However, the generation of modeling constructs—a key requirement in model transformation—requires a language that is more expressive than pure Datalog. Figure 6 shows a simplified version of the rule for translating SQL columns into the corresponding *GeRoMe* elements. As it is an equivalence rule, it can be used to import SQL schemas into *GeRoMe*, and to export *GeRoMe* models back into SQL schemas.

The first part of the rule refers to the elements of the SQL schema; the corresponding facts are generated by traversing the SQL schema. The second part of the rule specifies a fragment of the *GeRoMe* model: a model element with the identifier ID; an attribute role for this element; then, the aggregate role of the element TableID (specified by another rule) gets the attribute as an additional property; finally, the maximum cardinality of the attribute is set to one as attributes in the relational model are single-valued.

```
[
  sql_column(ID),
  sql_column_table(ID, TableID)
] <=> [
  modelElement(ID),
  attribute(ID),
  property(aggregate(TableID), attribute(ID)),
  max(attribute(ID), :val(1))
].
```

Fig. 6 Simplified rule for translating SQL columns to *GeRoMe* model elements

Our representation for the data instance level again extends ideas from Telos. In Telos, a base relation *proposition* expresses the relationships between objects: $P(o,x,l,y)$ states that the object x has a relationship with label l to object y. The variable o represents the object identifier of this link. We can make further statements about the type of link, e.g., whether it is an attribute link, an instantiation, or a specialization. With only a few predefined classes and only few constraints for this base relation, Telos supports the representation of models as well as of its instances, but also metamodels, metametamodels, …

In *GeRoMe*, an instance of a model is described by a set of objects; each object may have links to other objects (associations) or atomic values (attributes). In contrast to Telos, *GeRoMe* thus distinguishes objects and values. Furthermore, each object has a type (an instantiation link to a model element). Syntactically, *GeRoMe* instances can be represented as a set of logical facts using a limited set of predefined predicates. This enables the use of logical languages to express mappings on the instance level. Even if data instances have complex, nested structures, the Datalog representation just uses 'flat' facts.

The logical representation of *GeRoMe* uses the predicates *inst*, *attr*, *value*, and *part* to describe an instance of a model. The model is defined by a set of model elements M and an instance is described by a set of objects O. Furthermore, we have a set A of atomic values (literals).

- $inst(o,m)$ denotes that the object $o \in O$ is an instance of the model element $m \in$ M.
- $value(o,v)$ denotes that the object $o \in O$ has the value $v \in A$. This is only possible, if o is an instance of a model element that plays a domain role.
- $attr(o,a,o_v)$ denotes that the object $o \in O$ has an attribute of type $a \in$ M, and the value for this attribute is represented by the object $o_v \in O$.
- $part(o,ae,o_p)$ denotes that the object $o \in O$ is an association, and the object $o_p \in O$ is a participator in this association for the association end $ae \in$ M.

As the predicates *value* and *attr* are often used in combination for simple typed attributes, we use the predicate $av(o,a,v)$ as a shortcut to denote that the object o has a value v for the attribute a. As an example, Fig. 7 shows instances for the models in Fig. 5.

Concrete Instance	***GeRoMe* Instance**
car(1,76,45) *road*(7,766, "Ahornstr.")	$inst(t_1,car)$, $av(t_1,id,1)$, $av(t_1,speed,76)$, $av(t_1,roadId,7)$ $inst(t_2,road)$, $av(t_2,name$, "Ahornstr."), $av(t_2,roadId,7)$, $av(t_2,length,766)$

`<road id="7">`
`  <car id="1" speed="76"/>`
`</road>`
	$inst(x_r,root)$, $inst(x_0,road)$, $inst(x_1,roadT)$, $part(x_0,parent,x_r)$, $part(x_0,child,x_1)$, $av(x_1,id,45)$, $inst(x_2,car)$, $inst(x_3,carT)$, $part(x_2,parent,x_1)$, $av(x_3,id,1)$, $av(x_3,speed,76)$, $part(x_2,child,x_3)$

Fig. 7 Example instances for the models of Fig. 5

The description of the relational instance in *GeRoMe* is straightforward. For each tuple, there is a corresponding object (t_1 and t_2), and each attribute value of these tuples is defined by an *av* predicate. In the XML instance, we first define an object x_r for the (invisible) document root. Then, x_0 denotes the road element which is an association between the document root x_r and the instance x_1 of the complex type roadT. Attribute values are defined in the same way as in the relational instance. The instance of the car element x_2 is also an association; it links the parent element x_1 with the child element x_3, which is an instance of carT.

In practice, we never explicitly create instances of *GeRoMe* models in this verbose representation; it is only the formal basis for the definition of mappings. Our model-based code generation GeRoMeSuite transforms data during mapping execution directly from one native representation into another according to the specified mapping Kensche et al. 2007b.

4.4 Languages for schema mappings

The choice of GAV and LAV mappings mentioned in Section 2 is only one aspect of schema mappings. In addition, we have tasks like explicit data exchange or even physical generation of new data.

The choice of a mapping query language depends obviously on the choice for the metamodel. In the literature, the Relational Data Model is most frequently used for data integration systems, and a relational query language chosen for representing mappings. However, the full expressive power of a relationally complete query language (such as Relational Algebra) makes reasoning over mappings undecidable. Query containment (which has often to be proven during query rewriting) is only decidable for conjunctive, i.e. Select-Project-Join (SPJ) queries Shmueli 1993.

In Clio, Fagin et al. 2005a, b initially proposed to use tuple-generating dependencies (tgds) Beeri and Vardi 1984 for representing mappings between relational schemas, because tgds can be easily translated into executable queries.

A *source-to-target tuple-generating dependency* (*s-t tgd*) has the form:

$$\forall x(\varphi_S(x) \rightarrow \exists y \psi_T(x, \mathbf{y}))$$

x and y are sets of variables, φ_S and ψ_T are conjunctive queries over the relational schemas S and T, respectively. A mapping M is then defined as $M = <S,T,\Sigma>$ where S and T are the mapped schemas, and Σ is a set of s-t tgds. Their simplicity combined with reasonable expressive power is a strength of s-t tgds.

Clio Hernández et al. 2001 creates mappings over a nested relational model to support mappings between relational databases and XML data. Using a set of value correspondences as input, Clio is able to generate queries which transform source data into the desired target data structure. However, it would still be difficult to extend this mapping representation to express a mapping between other models, such as UML models, because there is simply no appropriate query language.

Another drawback of these *basic mappings* in Clio is pointed out in Fuxman et al. 2006: the mappings do not reflect the nested structure of the data. This leads to an inefficient execution of the mappings and redundant mapping specifications as parts of the mapping have to be repeated for different nesting levels. Furthermore, the desired grouping of the target data cannot be specified using basic mappings, as they would cause redundancy in the target. A nested mapping language introduced in Fuxman et al. 2006 addresses these problems. Furthermore, they provide an algorithm to

compute the nested mappings from simple morphisms which can be executed more efficiently than basic mappings.

Because executable mappings usually drive the transformation of instances of models, Melnik et al. Melnik et al. 2005 specified the semantics of model management operators by relating the instances of the operator's input and output models. They also implemented two model management system prototypes to study the specifying and manipulating of executable mappings. In the first implementation, they modified Rondo's Melnik et al. 2003b language to define path morphisms and showed that it is possible to generate executable mappings in a form of relational algebra expressions. On the positive side, this system works correctly whenever the input is specified using path morphisms, and the input is also closed under operators which return a single mapping. However, the expressiveness of path morphisms is very limited. To overcome this limitation, they developed a new prototype Melnik et al. 2005 in which mappings are specified using embedded dependencies. The expressiveness is improved, but it suffers from the problem that embedded dependencies are not closed under composition. Because of this problem, the output of the very important *Compose* operator cannot be represented as an embedded dependency; thus, a sequence of model management operators may not be executable.

4.4.1 Mapping composition

In general, the problem of composing mappings has the following definition: Given a mapping M_{12} from model S_1 to model S_2, and a mapping M_{23} from model S_2 to model S_3, derive a mapping M_{13} from model S_1 to model S_3 that is equivalent to the successive application of M_{12} and M_{23} Fagin et al. 2005a, b.

So far, mapping composition has been studied only for mappings which use the Relational Data Model as a basis Bernstein et al. 2006; Fagin et al. 2005a, b, MaHa03.

In Madhavan and Halevy 2003, the semantics of the *Compose* operator is defined relative to a class Q of queries over the model S_3. For every query $q \in$ Q, the certain answers for q wrt. M_{13} are the same as the certain answers for q wrt. M_{12} and M_{23}. This provided a solid basis for further research on mapping composition, but suffers from certain drawbacks caused by the fact that the semantics is defined relative to a class Q of queries. Fagin et al. 2005a, b proposed a different semantics which is defined over instance spaces of schema mappings. M_{13} is the composition of M_{12} and M_{23} if the instance space of M_{13} is the set-theoretic composition of the instance spaces of M_{12} and M_{23}. Under this semantics the mapping composition M_{13} is unique up to logical equivalence.

Another approach to define composition uses relational algebra expressions as mappings Bernstein et al. 2006. An incremental algorithm tries to replace as many symbols as possible from the "intermediate" model. Because the result of mapping composition cannot be always expressed as relational algebra expressions, the algorithm may fail under certain conditions; this limitation is in line with the results of Fagin et al. 2005a, b.

4.4.2 Towards composable, executable, generic schema mappings

Fagin et al. 2005a, b proved that the language of s-t tgds is not closed under composition. To illustrate the problem, we adapt an example from Fagin et al. 2005a, b to our traffic scenario. Suppose we have a schema S_1 with one relation $Travels_1(C,R)$ which means that a car with the

id C is traveling on a road named R. Another schema S_2 has two relations: $Travels_2(C,R)$ which is a copy of $Travels_1(C,R)$, and $Car_2(C,N)$ which states that a car C is owned by a person with name N. The mapping M_{12} between these schemas can be expressed using the following s-t tgds:

$$\forall c \forall r \; Travels_1(c,r) \quad \rightarrow Travels_2(c,r)$$

$$\forall c(\forall r \;\; Travels_1(c,r)) \;\; \rightarrow \exists n \;\; Car_2(c,n)$$

When we execute this mapping, transform data from S_1 to S_2, we do not have data for n as this information is not contained in S_1. However, we know that a car can be owned only by one person; thus, the value of n depends on the car id c. If we execute the mapping, we can thus create 'labeled null values' for n.

Assume there is a third schema S_3 with a single relation $DrivesOn_3(N,R)$ which holds information about persons driving on specific roads. The mapping M_{23} between S_2 and S_3 can be also expressed as a s-t tgd:

$$\forall n \forall r(\forall c \; Car_2(c,n) \wedge Travels_2(c,r)) \rightarrow DrivesOn_3(n,r)$$

If we now want to compose the mappings to a mapping $M_{13}=M_{12} \circ M_{23}$, a correct formula in first-order logic would be

$$\forall c \exists n \forall r \; Travels_1(c,r) \rightarrow DrivesOn_3(n,r) \tag{1}$$

However, this is not a valid s-t tgd as existential quantification is only allowed on the right-hand side of the implication. Note that the s-t tgd

$$\forall c \forall r \; Travels_1(c,r) \rightarrow \exists n \;\; DrivesOn_3(n,r)$$

is not a composition of M_{12} and M_{23} as n depends now on both, c and r, which is not correct with respect to composition semantics. To ameliorate the problem, we can skolemize formula (1) and replace n with a Skolem function $f(c)$:

$$\exists f(\forall c \forall r \; Travels_1(c,r) \rightarrow DrivesOn_3(f(c),r))$$

This is a second-order formula as we quantify over the function symbol f. Fagin et al. 2005a, b showed that second-order tgds (SO tgds) are the smallest class of formulas which can be used to represent the result of the composition of any finite s-t tgds. Thus, SO tgds are closed under composition, and mappings expressed as SO tgds can be executed in polynomial time. Therefore, SO tgds are a good formalization of mappings, albeit only for mapping relational schemas.

By extending SO tgds to *GeRoMe* we have enabled the definition of generic schema mappings Kensche et al. 2009. We will use the running example to illustrate that SO tgds and the logical representation of *GeRoMe* instances fit nicely together and enable generic mappings which are expressive, executable, and composable.

Suppose, we want to map the relational schema R from Fig. 4 to the target XML schema T, in which cars are nested into road elements. We can use the example instances in Fig. 7 as templates for the conjunctive queries that specify the mapping:

$$\begin{aligned}
&\forall I \forall S \forall R(\forall T_1 \forall T_2(\\
&\quad inst(T_1,\ car) \wedge av(T_1,\ id, I) \wedge av(T_1,\ speed,\ S) \wedge av(T_1,\ roadId,\ N) \wedge \\
&\quad inst(T_2,\ road) \wedge av(T_2,\ roadId,\ N)) \Rightarrow \\
&\qquad \exists Xr \exists X0 \exists X1 \exists X2 \exists X3 \\
&\qquad inst(Xr, root) \wedge inst(X0, road) \wedge \\
&\qquad part(X_0,\ parent, X_r) \wedge part(X_0, child, X_1) \wedge \\
&\qquad inst(X_1, roadType) \wedge av(X_1,\ id,\ N) \wedge inst(X_2,\ car) \wedge \\
&\qquad inst(X_3,\ carType) \wedge part(X_2,\ parent,\ X_1) \wedge part(X_2,\ child,\ X_3) \wedge \\
&\qquad av(X_3,\ id,\ I) \wedge av(X_3,\ speed,\ S))
\end{aligned}$$

The concrete values for attributes and object identifiers have been replaced with variables. As the target schema does not contain information about length and name of roads, we do not use the corresponding predicates on the source side. This formula is a valid s-t tgd, but it does not specify how to structure the data on the target side. In this case, there is one road element for each tuple in the road relation, as roadId is the key of the road relation and all cars driving on this road should be represented as nested elements. We cannot infer this constraint from the mapping definition above, as all object variables are existentially quantified, i.e. they can have different values for each matching pair of car and road tuples on the source side.

To overcome this problem, we can skolemize the formula in a similar way as in the relational example. The variables X_i on the target side will be replaced by corresponding functions $f_i(I,S,R)$. Note that I, S, and R are the arguments of these functions as they are the universally quantified variables which appear on both sides of the implication. The resulting formula is

$$\begin{aligned}
&\exists f_r \exists f_0 \exists f_1 \exists f_2 \exists f_3(\forall I \forall S \forall R(\forall T_1 \forall T_2 \\
&\quad inst(T_1,\ car) \wedge av(T_1,\ id,\ I) \wedge av(T_1,\ speed,\ S) \wedge av(T_1,\ roadId,\ N) \wedge \\
&\quad inst(T_2,\ road) \wedge av(T_2,\ roadId,\ N)) \Rightarrow \\
&\qquad inst(fr(I,S,R), root) \wedge inst(f0(I,S,R), road) \wedge \\
&\qquad part(f_0(I,S,R), parent, f_r(I,S,R)) \wedge part(f_0(I,S,R), child, f_1(I,S,R)) \wedge \\
&\qquad inst(f_1(I,S,R), roadType) \wedge av(f_1(I,S,R),\ id,\ N) \wedge inst(f_2(I,S,R),\ car) \wedge \\
&\qquad inst(f_3(I,S,R),\ carType) \wedge part(f_2(I,S,R),\ parent,\ f_1(I,S,R)) \wedge \\
&\qquad part(f_2(I,S,R), child,\ f_3(I,S,R)) \wedge av(f_3(I,S,R), id,\ I) \wedge \\
&\qquad av(f_3(I,S,R), speed, S)))
\end{aligned}$$

This is now a valid SO tgd, but it still would not allow generating correctly structured data: All object identifiers depend on I, S, and R, such that for each combination of values for these variables, there will be a road element with a nested car element. As I is the key for cars, a road element will be generated for each car, and this is not the desired structured.

To address this problem, we need to consider the schema information in the *GeRoMe* model to figure out the correct data structuring. We will see that the constraints allow only one instance of the root element for the whole document; thus, the corresponding Skolem function f_r should have no arguments (i.e., it is a constant). Roads are identified by roadId; thus, the

Skolem functions for the road element (f_0) and the road type (f_1) should have only R as argument. The car elements are identified by the car id I. Therefore, the Skolem functions for the instances of the car element (f_2) and the car type (f_3) have to include the variable I in their argument list. In addition, these functions need also the identifying variables of the parent element, as the car elements are nested under road elements. Consequently, the resulting functions are $f_2(I,R)$ and $f_3(I,R)$. The revised formula is given here:

$$
\begin{aligned}
&\exists f_r \exists f_0 \exists f_1 \exists f_2 \exists f_3 (\forall I \forall S \forall R (\forall T_1 \forall T_2 \\
&\quad inst(T_1,\ car) \wedge av(T_1,\ id,\ I) \wedge av(T_1,\ speed,\ S) \wedge av(T_1,\ roadId,\ N) \wedge \\
&\quad inst(T_2,\ road) \wedge av(T_2,\ roadId,\ N)) \Rightarrow \\
&\qquad inst(fr(), root) \wedge inst(f0(R), road) \wedge \\
&\qquad part(f_0(R), parent, f_r()) \wedge part(f_0(R), child, f_1(R)) \wedge \\
&\qquad inst(f_1(R), roadType) \wedge av(f_1(R), id, N) \wedge inst(f_2(I,R), car) \wedge \\
&\qquad inst(f_3(I,\ R), carType) \wedge part(f_2(I,R), parent,\ f_1(I,R)) \wedge \\
&\qquad part(f_2(I,R), child,\ f_3(I,R)) \wedge av(f_3(I,R), id, I) \wedge \\
&\qquad av(f_3(I,R),\ speed,\ S)))
\end{aligned}
$$

By using the generic mapping language and Skolem functions, we can define mappings between arbitrarily structured models in various modeling languages. To execute the mappings, we do not have to provide an interpretation of the Skolem functions. The Skolem functions will be used by the mapping compiler to generate appropriate queries or update statements which take the defined structure into account.

5 Schema matching and merging

The formal definition of GeRoMe and its mapping language enable the formal definition and verification of model management operators. We discuss briefly the realization of the operators *Match* and *Merge*. Schema merging and schema matching are two related operators, in that the output of matching can be used as input for schema merging.

5.1 Schema matching

Schema matching is the task of identifying a set of correspondences (also called a morphism) between schema elements. Many aspects have to be considered during the process of matching, such as data values, element names, constraint information, structure information, domain knowledge, cardinality relationships, and so on. All this information is useful in understanding the semantics of a schema, but it can be a very time consuming problem to collect and apply this information.

Therefore, automatic methods are required for schema matching Rahm and Bernstein 2001; Shvaiko and Euzenat 2005. *Element-Level Matchers* separately take the information of each schema element into account, using linguistic information (name of the element) or constraint information (data type, key constraints). *Structure-Level Matchers* use graph matching to measure the similarity of the structures implied the schema. *Instance-Level Matchers* employ data instances to match schema elements: If the instance sets of two elements are similar, or have a similar value distribution, this might indicate a similarity of the schema elements.

Machine-Learning Matchers use either instance data or previously identified matches as training data to detect similar matches in new schema matching problems.

It is widely agreed, that no single method can solve the schema matching problem. Therefore, matching frameworks such as COMA++ Aumueller et al. 2005, Protoplasm Bernstein et al. 2004, or YAM Duchateau et al. 2009 combine multiple individual matching methods to achieve a better result. For the heterogenous case, we have developed an extensible and flexible matching framework in our model management system *GeRoMeSuite* Kensche et al. 2007a, which is also able to combine several matching methods in entirely configurable matching strategies. We could show that our system in particular good for heterogeneous matching tasks, e.g., matching of XML schemas and OWL ontologies Quix et al. 2007. The field of ontology matching had more attention in the recent years than schema matching because of the structured evaluation of ontology matching systems in the Ontology Alignment Evaluation Initiative (OAEI, Euzenat et al. 2011).

Such logical methods include, for example, the validation of correspondences. A computed alignment should be consistent with the information present in the matched ontologies (or schemas). If it is possible to derive an inconsistency from a correspondence, the identified correspondences may be wrong. For example, in ASMOV Jean-Mary et al. 2009, validation of the computed alignment is a key concept in their ontology matching system and greatly improves the quality of the match result.

Our matching framework also incorporates validation methods for schema matching. We could show that such methods also improve the quality of matching for our generic approach (e.g., by comparing the results in Quix et al. 2008 and Quix et al. 2009). This also demonstrates another advantage of the rich generic metamodel: due to the exact representation of models in *GeRoMe*, we are able to apply these logical methods not only to ontologies, but to other modeling languages as well (e.g., by exploiting inheritance relationships or foreign key constraints).

Although the field of schema and ontology matching has made significant improvements in the recent years, there are still a lot of challenges to be addressed. For example, Shvaiko and Euzenat 2012 mention efficiency, matching with background knowledge, matcher selection and tuning as important requirements for ontology matching systems. We have developed a method for automatically retrieving background knowledge in form of ontologies from the web Quix et al. 2011. For a matching task, we inspect the source and target models and extract a few descriptive keywords from the models, in order to characterize the domain of the models to be matched. Using these keywords, we employ traditional search engines or specific ontology search engines (e.g., Swoogle Ding et al. 2004 or Watson d'Aquin and Motta 2011) to find ontologies on the web. In addition to matching the source and target models directly, we thus match the input models also with the background ontology. For example, the ontology O in Fig. 4 can be seen as an ontology bridging the semantic gap between the input models S and T. By composing the computed matches, we can then infer more matches between the models S and T and thereby get a more accurate mapping. In an extensive evaluation of the approach Quix et al. 2011, we could show that the 'noise' which might be introduced by inappropriate background knowledge is low and that it is more useful to use several ontologies as background knowledge instead of just one.

5.2 Schema merging

The challenging part in schema merging is a formal definition of the desired outcome. How can we characterize the requirements for the integrated schema formally? How can we prove that a schema integration method actually produced the correct result?

In Pottinger and Bernstein 2003, requirements for a *Merge* operator include the *preservation of the original semantics* of the input schemas (e.g., elements, relationships, and equalities) and the *minimality of the merged schema* (i.e., no extra information should be added). We present a method based on the previous formalisms that achieves these goals under certain assumptions.

In the previous sections, we have addressed extensional mappings between instances of schemas for tasks such as data translation or query rewriting. The semantics of extensional mappings (and their composition) is defined with respect to the instances of schemas.

For schema merging, we need *intensional mappings*. For example, consider relational databases from two different cities A and B which maintain information about roads, e.g., both databases have one relation of the form *road*(*id*, *length*, *name*). We could state the relationship between the two databases using the following mapping expression:

$$\forall I \forall L \forall N \; road_A(I, L, N) \rightarrow road_B(I, L, N).$$

This mapping is not correct if we consider the extensions of the relations. A road in city A is not a road in city B and vice versa. Nevertheless, at the intensional level, the mapping is useful because it states that the two relations have the same intended semantics. If additionally both databases would contain information for the same domain (i.e., for the same city), then the mapping would be correct. This difference in the semantics of mappings has been characterized in Calvanese et al. 1998.

Schema merging is about integrating models according to their intensional semantics. It has the goal to construct a "duplicate-free" union of the input models and mapping, with respect to the real world concepts described by the model elements. The integrated model should describe each real world concept only once. Thus, we need intensional mappings for schema integration.

In our schema merging approach Li et al. 2010; Li and Quix 2011, tuple-generating dependencies (tgds) are used to express the constraints of the input schemas as well as the inter-schema constraints (Fig. 8).

Assume we want to merge a set of schemas $S_1, \ldots, S_n$. We first construct S as the duplicate-free union of $S_1, \ldots, S_n$ (if there are elements with identical names, they need to be renamed). The mappings between the input schemas as well as the constraints are defined as tgds in the input mapping M_i. When we create the integrated schema T, we also produce two mappings: M_o is the output mapping responsible for translating data from the sources to the new target schema; M_r is a recovery mapping (or witness mapping) which is basically an inversion of M_o. The integrated schema T is created by a step-by-step procedure in which we remove incrementally elements from the input schema and check whether the obtained schema still fulfills the requirements of the integrated schema.

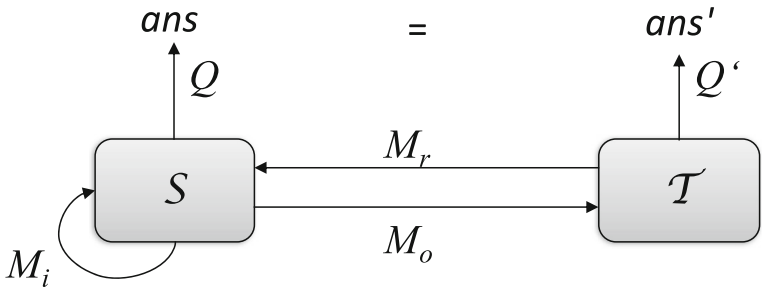

Fig. 8 The basic idea for Schema Merge using logical mappings

In our approach, the main requirement is that queries over the integrated schema should have the same result as over the source schemas. We can prove this property by using the generated mappings M_o and M_r. If we can prove that the answers for queries are the same when we evaluate them directly over S and when we evaluate them over S through the mapping M_o and M_r, then we know that no information has been lost in the integrated schema T.

This means that we must be able to prove query equivalence for all queries in a specific query language over the integrated schema and the source schemas. This is only possible if we have a formal mapping from the data sources to the integrated schema and vice versa. As query equivalence is decidable for conjunctive queries, we chose the class of conjunctive queries over relational schemas as our query language.

The target schema T is minimized by identifying elements which are potentially redundant. Redundant attributes are detected by reasoning over the input mappings and constraints. Our implementation of this merge method Li et al. 2011 also considers the case of collapsing relations, i.e., it removes redundant relations by discovering bi-directional inclusion dependencies Li et al. 2011.

6 Conclusion and outlook

In this paper, we have reviewed the evolution of data-centric (enterprise information integration) approaches that emphasize the semantics of information integration. We have shown that this approach has enabled significant progress in automation of many data and model management tasks, especially in the context of the relational data model and its extensions, e.g. to nested relations. With our *GeRoMe* approach, we have also illustrated current research on how to extend these approaches to the case of heterogeneous data models, without losing again the advantages of clear semantics and highly automated tools.

We intentionally limited our discussion to the case of structured or semi-structured data with a schema. The linkage of these approaches to text and media information integration which are also subject to intense research in the last years, still remains to be explored in depth.

But even within schema-based approaches, there is still a long way to go, in order to truly conquer the challenge of heterogeneity. While our algorithms in GeRoMeSuite address surprisingly well the aspects of (automatically) executable mapping in a significantly richer model language context than earlier solutions, and provide decent assistance for heterogeneous schema matching, our present solution for merging—information-preserving schema integration including target schema minimization—is unfortunately only provably correct and complete for the case of relational model integration; a tractable case where this also holds for heterogeneous merging remains an open problem.

References

Abiteboul, S., Hull, R., Vianu, V. (1995). Foundations of databases. Addison-Wesley.

Arenas, M., Barceló, P., Libkin, L., Murlak, F. (2010). Relational and XML data exchange. Synthesis lectures on data management. Morgan & Claypool Publishers.

Arens, Y., Knoblock, C. A., & Shen, W.-M. (1996). Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems, 6*(2–3), 99–130.

Atzeni, P., & Torlone, R. (1996). Management of multiple models in an extensible database design tool. In P. M. G. Apers, M. Bouzeghoub, & G. Gardarin (Eds.), *Proc. 5th international conference on extending database technology (EDBT)* (Lecture Notes in Computer Science, Vol. 1057, pp. 79–95). Avignon: Springer.

Atzeni, P., Cappellari, P., & Bernstein, P. A. (2005). A multilevel dictionary for model management. In L. M. L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos, & O. Pastor (Eds.), *Proc. 24th international conference on conceptual modeling (ER)* (Lecture Notes in Computer Science, Vol. 3716, pp. 160–175). Klagenfurt: Springer.

Atzeni, P., Cappellari, P., & Bernstein, P. A. (2006). Model-independent schema and data translation. In Y. E. Ioannidis, M. H. Scholl, J. W. Schmidt, F. Matthes, M. Hatzopoulos, K. B¨ohm, A. Kemper, T. Grust, & C. Böhm (Eds.), *Proc. 10th international conference on extending database technology (EDBT)* (Lecture Notes in Computer Science, Vol. 3896, pp. 368–385). Munich: Springer.

Atzeni, P., Bellomarini, L., Bugiotti, F., & Gianforme, G. (2009). Mism: A platform for model-independent solutions to model management problems. *Journal of Data Semantics, 14*, 133–161.

Aumueller, D., Do, H. H., Massmann, S., & Rahm, E. (2005). Schema and ontology matching with COMA++. In F. Ozcan (Ed.), *Proceedings of the ACM SIGMOD international conference on management of data* (pp. 906–908). Baltimore: ACM.

Bachman, C.W., Daya, M. (1977). The role concept in data models. In: Proceedings of the Third International Conference on Very Large Data Bases (VLDB), pp. 464–476. IEEE-CS and ACM, Tokyo, Japan.

Batini, C., Lenzerini, M., & Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys, 18*(4), 323–364.

Baumeister, M., & Jarke, M. (1999). *Compaction of large class hierarchies in databases for chemical engineering. proceedings 8. gi-fachtagung für datenbanksysteme in büro, technik und wissenschaft (BTW)* (pp. 343–361). Freiburg: Springer.

Beeri, C., & Vardi, M. Y. (1984). A proof procedure for data dependencies. *Journal of the ACM, 31*(4), 718–741.

Bergamaschi, S., Castano, S., Vincini, M., & Beneventano, D. (2001). Semantic integration of heterogeneous information sources. *Data & Knowledge Engineering, 36*(3), 215–249.

Bernstein, P. A., & Haas, L. M. (2008). Information integration in the enterprise. *Communications of the ACM, 51*(9), 72–79.

Bernstein, P. A., & Melnik, S. (2007). Model management 2.0: Manipulating richer mappings. In L. Zhou, T. W. Ling, & B. C. Ooi (Eds.), *Proc. ACM SIGMOD intl. conf. on management of data* (pp. 1–12). Beijing: ACM Press. doi:10.1145/1247480.1247482.

Bernstein, P. A., Halevy, A. Y., & Pottinger, R. (2000). A vision for management of complex models. *SIGMOD Record, 29*(4), 55–63.

Bernstein, P. A., Melnik, S., Petropoulos, M., & Quix, C. (2004). Industrialstrength schema matching. *SIGMOD Record, 33*(4), 38–43.

Bernstein, P.A., Green, T.J., Melnik, S., Nash, A. (2006). Implementing mapping composition. In: U. Dayal, K.Y. Whang, D.B. Lomet, G. Alonso, G.M. Lohman, M.L. Kersten, S.K. Cha, Y.K. Kim (eds.) Proc. 32nd Intl. Conference on Very Large Data Bases (VLDB), pp. 55–66. ACM Press.

Biskup, J., & Convent, B. (1986). A formal view integration method. In C. Zaniolo (Ed.), *Proc. ACM SIGMOD intl. conf. on management of data* (pp. 398–407). Washington: ACM Press.

Brandt, S. C., Morbach, J., Miatidis, M., Theißen, M., Jarke, M., & Marquardt, W. (2008). An ontology-based approach to knowledge management in design processes. *Computers & Chemical Engineering, 32*(1–2), 320–342.

Brodie, M.L. (2010). Data integration at scale: From relational data integration to information ecosystems. In: Proc. 24th IEEE Intl. Conf. on Advanced Information Networking and Applications (AINA), pp. 2–3. IEEE Computer Society, Perth, Australia.

Calì, A., Calvanese, D., Giacomo, G. D., & Lenzerini, M. (2004). Data integration under integrity constraints. *Information Systems, 29*(2), 147–163. doi:10.1016/S0306-4379(03)00050-4.

Calvanese, D., Giacomo, G. D., Lenzerini, M., Nardi, D., & Rosati, R. (1998). Description logic framework for information integration. In A. G. Cohn, L. K. Schubert, & S. C. Shapiro (Eds.), *Proceedings of the sixth international conference on principles of knowledge representation and reasoning (KR'98)* (pp. 2–13). Trento: Morgan Kaufmann.

Calvanese, D., Giacomo, G. D., Lenzerini, M., Nardi, D., & Rosati, R. (2001). Data Integration in Data Warehousing. *International Journal of Cooperative Information Systems (IJCIS), 10*(3), 237–271.

Casanova, M.A., Vidal, V.M.P. (1983). Towards a sound view integration methodology. In: Proc. 2nd ACM Symposium on Principles of Database Systems (PODS), pp. 36–47. ACM, Atlanta, GA.

Cattell, R. (2010). Scalable SQL and NoSQL data stores. *SIGMOD Record, 39*(4), 12–27.

Ceri, S., Pelagatti, G. (1984). Distributed databases: principles and systems. McGraw-Hill Book Company.

Collet, C., Huhns, M. N., & Shen, W. M. (1991). Resource integration using a large knowledge base in carnot. *IEEE Computer, 24*(12), 55–62.

d'Aquin, M., & Motta, E. (2011). Watson, more than a semantic web search engine. *Semantic Web Journal, 2*(1), 55–63. http://www.semantic-web-journal.net/content/new-submission-watson-more-semantic-web-search-engine.

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R., Peng, Y., Reddivari, P. et al. (2004). Swoogle: a search and metadata engine for the semantic web. In: Proc. CIKM.

Do, H.H., Rahm, E. (2002). Coma -a system for flexible combination of schema matching approaches. In: Proc. 28th Intl. Conference on Very Large Data Bases (VLDB), pp. 610–621. Morgan Kaufmann, Hong Kong, China.

Dolk, D.R. (1988). Model management and structured modeling: the role of an information resource dictionary system. Communications of the ACM 31(6).

Duchateau, F., Coletta, R., Bellahsene, Z., & Miller, R. J. (2009). (Not) yet another matcher. In D. W. L. Cheung, I. Y. Song, W. W. Chu, X. Hu, & J. J. Lin (Eds.), *Proc. 18th ACM conference on information and knowledge management (CIKM)* (pp. 1537–1540). Hong Kong: ACM.

Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., & dos Santos, C. T. (2011). Ontology alignment evaluation initiative: 6 years of experience. *Journal on Data Semantics, 15*, 158–192.

Fagin, R., Kolaitis, P., Miller, R. J., & Popa, L. (2005a). Data exchange: Semantics and query answering. *Theoretical Computer Science, 336*, 89–124.

Fagin, R., Kolaitis, P. G., Popa, L., & Tan, W. C. (2005b). Composing schema mappings: Second-order dependencies to the rescue. *ACM Transactions on Database Systems, 30*(4), 994–1055.

Fagin, R., Haas, L.M., Hern'andez, M.A., Miller, R.J., Popa, L., Velegrakis, Y. (2009). Clio: Schema mapping creation and data exchange. In: A. Borgida, V.K. Chaudhri, P. Giorgini, E.S.K. Yu (eds.) Conceptual Modeling: Foundations and Applications, Lecture Notes in Computer Science, vol. 5600, pp. 198–236. Springer.

Fuxman, A., Hernández, M.A., Ho, C.T.H., Miller, R.J., Papotti, P., Popa, L. (2006). Nested mappings: Schema mapping reloaded. In: U. Dayal, K.Y. Whang, D.B. Lomet, G. Alonso, G.M. Lohman, M.L. Kersten, S.K. Cha, Y.K. Kim (eds.) Proc. 32nd Intl. Conference on Very Large Data Bases (VLDB), pp. 67–78. ACM Press.

Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J. D., et al. (1997). The tsimmis approach to mediation: Data models and languages. *Journal of Intelligent Information Systems, 8*(2), 117–132.

Geisler, S., Quix, C., Schiffer, S., & Jarke, M. (2012). An evaluation framework for traffic information systems based on data streams. *Transportation Research Part C, 23*, 29–55.

Genesereth, M. R., Keller, A. M., & Duschka, O. M. (1997). Infomaster: An information integration system. In J. Peckham (Ed.), *Proceedings of the ACM SIGMOD international conference on management of data* (pp. 539–542). Tucson: ACM Press.

Haas, L. M. (2007). Beauty and the beast: The theory and practice of information integration. In T. Schwentick & D. Suciu (Eds.), *ICDT, lecture notes in computer science* (Vol. 4353, pp. 28–43). Barcelona: Springer.

Haas, L. M., Hernández, M. A., Ho, H., Popa, L., & Roth, M. (2005). Clio grows up: From research prototype to industrial tool. In F. Ozcan (Ed.), *Proceedings of the ACM SIGMOD international conference on management of data* (pp. 805–810). Baltimore: ACM.

Halevy, A. Y. (2001). Answering queries using views: A survey. *VLDB Journal, 10*(4), 270–294.

Halevy, A. Y., Ives, Z. G., Madhavan, J., Mork, P., Suciu, D., & Tatarinov, I. (2004). The piazza peer data management system. *IEEE Transactions on Knowledge and Data Engineering, 16*(7), 787–798. doi:10.1109/TKDE.2004.1318562.

Haslhofer, B., Klas, W. (2010). A survey of techniques for achieving metadata interoperability. ACM Comput. Surv. 42(2).

Hernández, M.A., Miller, R.J., Haas, L.M. (2001). Clio: A semi-automatic tool for schema mapping. In: Proc. ACM SIGMOD Intl. Conference on the Management of Data, p. 607. ACM Press, Santa Barbara, CA.

ISO/IEC (1990). Information technology—Information Resource Dictionary System (IRDS) framework. International Standard ISO/IEC 10027:1990, ISO International Organization for Standardization.

ISO/IEC (2005). Information technology -Meta Object Facility (MOF). International Standard ISO/IEC 19502: 2005, ISO International Organization for Standardization.

Jarke, M., Gallersdörfer, R., Jeusfeld, M. A., & Staudt, M. (1995). ConceptBase -a deductive object base for meta data management. *Journal of Intelligent Information Systems, 4*(2), 167–192.

Jarke, M., Jeusfeld, M. A., Quix, C., & Vassiliadis, P. (1999). Architecture and Quality in Data Warehouses: An Extended Repository Approach. *Information Systems, 24*(3), 229–253.

Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P. (eds.) (2003). Fundamentals of data warehouses, 2 edn. Springer-Verlag.

Jarke, M., Jeusfeld, M., Nissen, H., Quix, C., Staudt, M. (2009). Metamodelling with datalog and classes: Conceptbase at the age of 21. In: Proc. 2nd Intl. Conf. Object Databases (ICOODB 09), pp. 95–112. Springer-Verlag.

Jean-Mary, Y. R., Shironoshita, E. P., & Kabuka, M. R. (2009). Ontology matching with semantic verification. *Journal of Web Semantics, 7*(3), 235–251.

Jeusfeld, M.A. (1992). Änderungskontrolle in deduktiven Objektbanken. PhD thesis, Universität Passau.

Jeusfeld, M. A., & Johnen, U. A. (1995). An executable meta model for reengineering of database schemas. *Intl. Journal of Cooperative Information Systems, 4*(2–3), 237–258.

Kensche, D., Quix, C., Chatti, M.A., Jarke, M. (2007). GeRoMe: A generic role based metamodel for model management. Journal on Data Semantics VIII, 82–117.

Kensche, D., Quix, C., Li, X., Li, Y. (2007). GeRoMeSuite: A system for holistic generic model management. In: C. Koch, J. Gehrke, M.N. Garofalakis, D. Srivastava, K. Aberer, A. Deshpande, D. Florescu, C.Y. Chan, V. Ganti, C.C. Kanne, W. Klas, E.J. Neuhold (eds.) Proceedings 33rd Intl. Conf. on Very Large Data Bases (VLDB), pp. 1322–1325. Vienna, Austria.

Kensche, D., Quix, C., Li, X., Li, Y., & Jarke, M. (2009). Generic schema mappings for composition and query answering. *Data & Knowledge Engineering, 68*(7), 599–621. doi:10.1016/j.datak.2009.02.006.

Kerner, B., Demir, C., Herrtwich, R., Klenov, S., Rehborn, H., Aleksic, M. et al. (2005). Traffic state detection with floating car data in road networks. In: Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems, pp. 700–705. Daimler Chrysler AG.

Kirk, T., Levy, A.Y., Sagiv, Y., Srivastava, D. (1995). The Information Manifold. In: Proceedings of the AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, pp. 85–91.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In L. Popa (Ed.), *Proc. 21st ACM symposium on principles of database systems (PODS)* (pp. 233–246). Madison: ACM Press. doi:10.1145/543613.543644.

Li, X., & Quix, C. (2011). Merging relational views: A minimization approach. In M. A. Jeusfeld, L. M. L. Delcambre, & T. W. Ling (Eds.), *Proc. 30th intl. conference on conceptual modeling (ER 2011)* (Lecture Notes in Computer Science, Vol. 6998, pp. 379–392). Brussels: Springer.

Li, X., Quix, C., Kensche, D., & Geisler, S. (2010). Automatic schema merging using mapping constraints among incomplete sources. In J. Huang, N. Koudas, G. J. F. Jones, X. Wu, K. Collins-Thompson, & A. An (Eds.), *Proc. 19th ACM conf. on information and knowledge management (CIKM)* (pp. 299–308). Toronto: ACM.

Li, X., Quix, C., Kensche, D., Geisler, S., & Guo, L. (2011). Automatic generation of mediated schemas through reasoning over data dependencies. In S. Abiteboul, K. B¨ohm, C. Koch, & K. L. Tan (Eds.), *Proc. 27th intl. conf. on data engineering (ICDE)* (pp. 1280–1283). Hannover: IEEE Computer Society.

Madhavan, J., & Halevy, A. Y. (2003). Composing mappings among data sources. In J. C. Freytag, P. C. Lockemann, S. Abiteboul, M. J. Carey, P. G. Selinger, & A. Heuer (Eds.), *Proc. of 29th intl. conference on very large data bases (VLDB)* (pp. 572–583). Berlin: Morgan Kaufmann.

Melnik, S., Rahm, E., & Bernstein, P. A. (2003a). Developing metadata-intensive applications with Rondo. *Journal of Web Semantics, 1*(1), 47–74.

Melnik, S., Rahm, E., Bernstein, P.A. (2003). Rondo: A programming platform for generic model management. In: Proc. ACM SIGMOD Intl. Conference on Management of Data, pp. 193–204. ACM, San Diego, CA.

Melnik, S., Bernstein, P. A., Halevy, A. Y., & Rahm, E. (2005). Supporting executable mappings in model management. In F. Ozcan (Ed.), *Proceedings of the ACM SIGMOD international conference on management of data* (pp. 167–178). Baltimore: ACM.

Miller, R. J., Haas, L. M., & Hernández, M. A. (2000). Schema mapping as query discovery. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, & K. Y. Whang (Eds.), *Proc. 26th intl. conference on very large data bases (VLDB)* (pp. 77–88). Cairo: Morgan Kaufmann.

Mork, P., Bernstein, P.A., Melnik, S. (2007). Teaching a schema translator to produce o/r views. In: Proc. 26th Intl. Conf. on Conceptual Modeling (ER'07), LNCS, vol. 4801, pp. 102–119. Springer.

Mylopoulos, J., Borgida, A., Jarke, M., & Koubarakis, M. (1990). Telos: Representing Knowledge About Information Systems. *ACM Transactions on Information Systems, 8*(4), 325–362.

Nardi, D., Brachman, R.J. (2003). An introduction to description logics. In: F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider (eds.) Description Logic Handbook. Cambridge University Press.

Nissen, H. W., & Jarke, M. (1999). Repository Support for Multi-Perspective Requirements Engineering. *Information Systems, 24*(2), 131–158.

Parent, C., & Spaccapietra, S. (1998). Issues and approaches of database integration. *Communications of the ACM, 41*(5), 166–178.

Pottinger, R., & Bernstein, P. A. (2003). Merging models based on given correspondences. In J. C. Freytag, P. C. Lockemann, S. Abiteboul, M. J. Carey, P. G. Selinger, & A. Heuer (Eds.), *Proc. of 29th intl. conference on very large data bases (VLDB)* (pp. 826–873). Berlin: Morgan Kaufmann.

Pottinger, R., & Halevy, A. Y. (2001). Minicon: A scalable algorithm for answering queries using views. *VLDB Journal, 10*(2–3), 182–198.

Quix, C. (2009). Meta data repository. In: L. Liu, M.T. Ozsu (eds.) Encyclopedia of Database Systems, pp. 1718–1722. Springer.

Quix, C., Kensche, D., & Li, X. (2007). Matching of ontologies with xml schemas using a generic metamodel. In R. Meersman & Z. Tari (Eds.), *Proc. OTM confederated international conf. CoopIS/DOA/ODBASE/GADA/IS* (Lecture Notes in Computer Science, Vol. 4803, pp. 1081–1098). Vilamoura: Springer.

Quix, C., Geisler, S., Kensche, D., Li, X.: Results of GeRoMesuite for OAEI 2008. In: Proc. 3rd Intl. Workshop On Ontology Matching (OM2008) (2008). URL http://data.semanticweb.org/workshop/om/2008/paper/main/13.

Quix, C., Geisler, S., Kensche, D., & Li, X. (2009). Results of geromesuite for oaei 2009. In P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, N. F. Noy, & A. Rosenthal (Eds.), *Proc. 4th intl. workshop on ontology matching* (CEUR Workshop Proceedings, Vol. 551). Chantilly: CEUR-WS.org.

Quix, C., Roy, P., Kensche, D. (2011). Automatic selection of background knowledge for ontology matching. In: Proc. Intl. Workshop on Semantic Web Information Management (SWIM), pp. 5:1–5:7. ACM, New York, NY, USA.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *VLDB Journal, 10*(4), 334–350.

Ramesh, B., & Jarke, M. (2001). Toward Reference Models of Requirements Traceability. *IEEE Transactions on Software Engineering, 27*(1), 58–93.

Richardson, J., Schwarz, P. (1991). Aspects: extending objects to support multiple, independent roles. In: Proc. ACM SIGMOD Intl. Conference on Management of Data, pp. 298–307. Denver, CO.

Shmueli, O. (1993). Equivalence of datalog queries is undecidable. *Journal of Logic Programming, 15*(3), 231–241.

Shu, N. C., Housel, B. C., Taylor, R. W., Ghosh, S. P., & Lum, V. Y. (1977). EXPRESS: A Data EXtraction, Processing, amd REStructuring System. *ACM Transactions on Database Systems, 2*(2), 134–174.

Shvaiko, P., Euzenat, J. (2005). A survey of schema-based matching approaches. Journal on Data Semantics IV, 146–171. LNCS 3730.

Shvaiko, P., Euzenat, J. (2012). Ontology matching: State of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering. To appear, preprint available at http://www.dit.unitn.it/~p2p/RelatedWork/Matching/SurveyOMtkde_SE.pdf.

Singh, M. P., Cannata, P. E., Jacobs, N., Ksiezyk, T., Ong, K., Sheth, A. P., et al. (1997). The carnot heterogeneous database project: Implemented applications. *Distributed and Parallel Databases, 5*(2), 207–225.

Smith, M. (2007). Toward enterprise information integration. Software Magazine. URL http://www.softwaremag.com/content/ContentCT.aspP=3034.

Spaccapietra, S., & Parent, C. (1994). View integration: A step forward in solving structural conflicts. *IEEE Transactions on Knowledge and Data Engineering, 6*(2), 258–274.

Staudt, M., & Jarke, M. (2000). View Management Support in Advanced Knowledge Base Servers. *Journal Intelligent Information Systems, 15*(3), 253–285.

Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM, 53*(4), 10–11.

Stubing, H., Bechler, M., Heussner, D., May, T., Radusch, I., Rechner, H., et al. (2010). simtd: A car-to-x system architecture for field operational tests. *IEEE Communications Magazine, 48*(5), 148–154.

Vogels, W. (2007). Data access patterns in the amazon.com technology platform. In: C. Koch, J. Gehrke, M.N. Garofalakis, D. Srivastava, K. Aberer, A. Deshpande, D. Florescu, C.Y. Chan, V. Ganti, C.C. Kanne, W. Klas, E.J. Neuhold (eds.) Proceedings 33rd Intl. Conf. on Very Large Data Bases (VLDB), p. 1. Vienna, Austria.

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer, 25*(3), 38–49.

Wiederhold, G (ed.). (1996). Special Issue on Intelligent Integration of Information. Journal of Intelligent Information Systems 6(2–3), 93–291.

Wong, R. K., Chau, H. L., & Lochovsky, F. H. (1997). A data model and semantics of objects with dynamic roles. In A. Gray & P. A. Larson (Eds.), *Proceedings of the 13th international conference on data engineering (ICDE)* (pp. 402–411). Birmingham: IEEE Computer Society.

Zhou, G., Hull, R., & King, R. (1996). Generating data integration mediators that use materialization. *Journal of Intelligent Information Systems, 6*(2–3), 199–221.