Modeling Data Federations in ORM

Herman Balsters[1] and Terry Halpin[2]

[1] University of Groningen, The Netherlands
e-mail: H.Balsters@rug.nl
[2] Neumont University, Utah, USA.
e-mail: terry@neumont.edu

Abstract: Two major problems in constructing data federations (for example, data warehouses and database federations) concern achieving and maintaining consistency and a uniform representation of the data on the global level of the federation. The first step in creating uniform representations of data is known as data extraction, whereas data reconciliation is concerned with resolving data inconsistencies. Our approach to constructing a global conceptual schema as the result of integrating a collection of (semantically) heterogeneous component schemas is based on the concept of exact views. We show that a global schema constructed in terms of exact views integrates component schemas in such a way that the global schema is populated by exactly those instances allowed by the local schemas (and in special cases, also the other way around). In this sense, the global schema is equivalent to the set of component schemas from which the global schema is derived. This paper describes a modeling framework for data federations based on the Object-Role Modeling (ORM) approach. In particular, we show that we can represent exact views within ORM, providing the means to resolve in a combined setting data extraction and reconciliation problems on the global level of the federation.

1 Introduction

Information systems composed of multiple, local systems that remain autonomous while sharing at least some of their information are called federated information systems. If the component systems are all databases, we speak of a *federated database system* ([33]). This paper addresses the situation where the component systems are pre-existing but are to interoperate in an integrated single framework.

Information integration is a complex problem relevant in fields such as data reengineering, data warehousing, web information systems, and service oriented architecture. Data integration systems include a global schema and a set of local schemas. Three approaches to data integration include: global-as-view (GAV) in which the global schema is defined in terms of the local schemas; local-as-view (LAV) in which local schemas are defined as views over the global schema; and data exchange, where both local and global schemas are pre-existing and mappings between them are then defined (cf. [26]). An overview of data integration concentrating on LAV and GAV can be found in [24]. Papers [1, 15] focus on LAV, and [4, 5, 34] focus on GAV. Our paper focuses on GAV in the context of database federations.

We address two major problems in this paper: semantic heterogeneity ([7, 24]), and the closed world assumption (CWA). Semantic heterogeneity arises when different meanings are assigned to shared data terms. The CWA assumes complete knowledge of relevant facts (propositions not stored in or derivable from the database are assumed to be false). Following the steps of extracting and creating uniform representations of data, the data reconciliation process is concerned with resolving data inconsistencies. GAV as a means to tackle semantic heterogeneity in database federations is described in [4, 5, 9, 34]. Paper [9] treats data integration under global integrity constraints; paper [34] concerns integration of local integrity constraints; Papers [4, 5] differ from [9] by also taking local integrity constraints into account, and by adopting exact views ([2, 4, 5]) instead of merely sound views ([9]). Exact views enable one to maintain the CWA on the global level, when the CWA is satisfied on the local levels. Paper [5] offers an improved algorithm for integrating local constraints on the global level, and also encompasses a far larger class of local constraints.

The results in [4, 5] were described in terms of the Unified Modeling Language (UML) [28] and its Object Constraint Language (OCL) [29, 35] in the context of database design [8,11, 12]. This paper, in contrast, describes a modeling framework for data federations based on the Object-Role Modeling (ORM) approach, also known as fact-oriented modeling. ORM is a conceptual approach for modeling, transforming, and querying information in terms of the underlying facts of interest, where facts and rules may be verbalized in language readily understandable by non-technical users of the business domain [26]. In contrast to Entity-Relationship (ER) modeling [10] and Unified Modeling Language (UML) class diagrams [28], ORM models are attribute-free, treating all facts as relationships (unary, binary, ternary etc.). ORM includes procedures for mapping to attribute-based structures, such as those of ER or UML. While this paper uses the notation of second generation ORM (ORM 2 [20]), we use the term "ORM" to include a number of closely related dialects, such as Natural language Information Analysis Method (NIAM) and Fully-Communication Oriented Information Modeling (FCO-IM). For a recent overview of ORM see [21], for a thorough treatment see [16]. For a comparison of ORM with UML see [18].

In the specific context of modeling data integration, ORM offers advantages over UML by facilitating the representation of exact views via easily understood objectified associations, and enabling the resolution in a combined setting of data extraction and reconciliation problems on the global level of the federation. ORM models also offer greater semantic stability, assisting with sustainability of the integration result. Data warehouses can be seen as examples of a special kind of data federations, where one restricts oneself to query-only applications. Recent publications in the context of modeling data warehouses, indicate that sustainability is an essential issue in the initial phases of the design ([14, 27, 30,32]).

The rest of this paper is structured as follows. Section 2 focuses on the problems of inconsistency and incompleteness. Section 3 considers semantic heterogeneity. Section 4 examines the integrated model for our running example. Section 5 discusses integration by ψ-maps and exact views. Section 6 compares our ORM approach to addressing data federation problems with that of UML/OCL. Section 7 summarizes the main results, suggests topics for further research, and lists references.

2 The Problems of Inconsistency and Incompleteness

Schema integration has to satisfy certain completeness and consistency requirements in order to reflect correct semantics of the different local schemata on the global integrated level [6]. In particular, each object on the local level should correspond to one object or object combination on the global level, and each global object should correspond to exactly one combination of local objects. Both requirements can be satisfied only if there exists an adequate mapping from the federated database states to the component database states. In [4, 5] such a mapping was termed a *ψ-map*, where *ψ* (psi) is an acronym for *p*reservation of *s*ystem *i*ntegrity.

Constructing a ψ-map can be challenging. First of all, the process of *data extraction* (cf. [4]) can give rise to inconsistencies related to the *ontologies* (cf. [31]) of the component databases. Matters such as naming conflicts (e.g. homonyms and synonyms), conflicts due to different underlying data types of attributes and/or scaling, and missing attributes all deal with differences in structure and semantics of the different local databases. By employing techniques such as renaming, conversion functions, default values, and addition of suitable extra attributes once can construct a common data model in which these (quasi-)inconsistencies are resolved.

A ψ-map, however, also has to ensure enforcement of both local integrity constraints (e.g. functional dependencies) on local database states, and global integrity constraints on federated database states. Hence, a ψ-map has to deal with the *data reconciliation* problem pertaining to the real inconsistencies due to conflicting integrity constraints. Following [4, 5, 34], we explain these inconsistencies using the terms local and global understandability.

In databases, *transparency* means that users need not see the internals of a database, e.g. the location of data on a disk. In the context of federated schemata, *global transparency* requires that global users need not see the local schemata or the global schema. At the global level, *global understandability* demands that global transactions (updates, queries) are accepted if they satisfy the global integrity constraints. *Local understandability*, on the other hand, demands that local transactions are accepted if the local integrity constraints are satisfied in the local component database. In [5] it was shown, given an arbitrary collection of component schemas, how to construct a corresponding federated schema and a ψ-map linking the local schemas and the federated schema, thus ensuring global understandability. Local understandability, however, is generally not feasible due to the general character that federation constraints can have. Should there be no extra purely federated constraints on the global level, then local understandability can be ensured.

A ψ-map can be used to define a so-called *exact view*. A global schema constructed in terms of exact views integrates component schemas in such a way that the global schema populates exactly those instances allowed by the local schemas (and in special cases, also conversely). In this sense, the global schema is equivalent to the set of component schemas from which the global schema is derived.

In contrast to the UML/OCL approach in [4, 5], this paper uses ORM to represent exact views in terms of objectified relations, without needing to introduce (often complex) derived classes or association classes, as would be the case in UML. The resulting ORM specification is easy to read, compared with OCL-style specifications.

3 Component Databases and Semantic Heterogeneity

Suppose we wish to integrate a customer relationship management (CRM) database and a sales database. ORM schemas for fragments of these databases are shown in Fig. 1. For simplicity, these schemas assume we are not interested in knowing which persons or employees are clients (so these are modelled as separate top-level types).

The two schemas are related (e.g. the object type Client has the same extension in both). Our aim is to create a global schema that integrates the two local schemas. In practice, each local schema may be part of a much larger local schema, but we assume that the local schemas shown here depict the fragments that the autonomous local schemas are willing to share in the global federation.

The problems faced when trying to integrate relational database schemas are well-known ([7, 9, 32]). We treat these problems at a conceptual level using ORM, focusing initially on *semantic heterogeneity* ([7, 24]), or differences in meaning attached to the shared data. We categorize this general problem into five main subproblems: homonyms/synonyms, data conversion, default values, missing attributes, and subtyping. These five categories form the main problems in integrating data. We now shortly describe in informal terms how these problems can be tackled, and subsequently show how these problems can be treated in the ORM-framework.

Conflicts due to homonyms are resolved by mapping two same name occurrences with different semantics to different names in the integrated model. For example, suppose that CRM.Salary means annual salary and Sales.salary means monthly salary. These must be given different names in the global schema. Synonyms are treated analogously, by mapping two different names (with the same semantics) to one common global name. We use the abbreviations **hom** (**syn**) to indicate that we have applied this method to solve a particular case of homonym (synonym) conflicts.

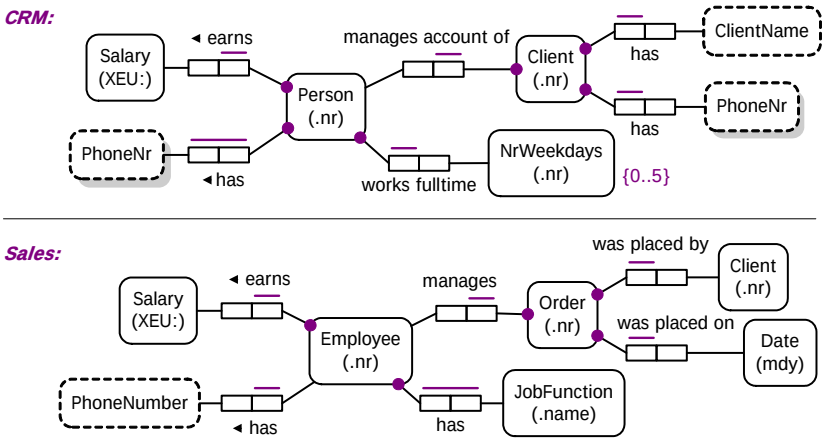

Fig. 1. ORM schemas for local CRM and Sales databases

Conflicts due to conversion arise when semantically related types are represented in different units or at a different granularity. For example, the two salary types have different temporal granularities (yearly and monthly). One solution is to convert each salary to a single standard salary in the global schema using a derivation rule for the conversion. For example, if we treat monthly salary as the standard the annual salary figures may be converted by multiplying by a factor 12 (in practice, the rule is more complex than this). Another kind of conversion can occur when the combination of types in one model has the same meaning as one type in another model. We use the abbreviation **conv** to indicate that we have applied a conversion method.

Conflicts due to default values occur when an attribute in only one local schema could be added to the second local schema by using a suitable default value. For example, we could add the fact type Employee works fulltime NrWeekdays to the Sales schema by stipulating a default value of 5 (indicating full-time employment) for the role played here by NrWeekdays. We use the abbreviation **def** to indicate that we have applied this method to solve a particular case of a default conflict.

Conflicts due to differentiation occur when an identification scheme that works locally fails in the global text. For example, suppose that the CRM and sales departments each issue employee numbers that are identifying within their department, but not globally (e.g. a person with employee number '3' in the CRM department might not be identical to a person with employee number '3' in the Sales department). In the global schema, employee number is no longer identifying. Also if someone may work for both departments, they can have different employee numbers in each. One way to resolve this problem in the global schema is to use a combination of department and the local employee number to identify employees. Typically it helps to also introduce a simple, global identifier and retain the compound identifier as a secondary reference scheme. This also makes it easier to match individuals with different local identifiers. We use the abbreviation **diff** to indicate that we have applied this method to solve a particular case of a differentiation conflict.

Resolution of such differentiation conflicts typically goes hand in hand with introduction of appropriate subtypes, to retain the constraints that certain details are maintained only for certain types from a local schema. Applying the method of adding new subtypes in the integration process is indicated by **sub**.

In the next section we illustrate how to apply the methods **hom, syn**, **conv**, **def**, **diff** and **sub** in the ORM-framework.

4 Integrating the Local Schemas

The Global1 ORM schema in Fig. 2 shows one way to integrate the local CRM and Sales schemas, which come from two different departments. It is possible for the same person to work for both departments, so the types CRM.Person and Sales.Employee overlap. We call the union of these types Global1.Employee, and introduce a simple global identifier (id) for it. The schema provides one way of retaining the local identification schemes for employees, by introducing the employment fact type, objectified as Employment, and declaring the external uniqueness constraint (circled bar) shown (so within a given department, employee numbers do identify).

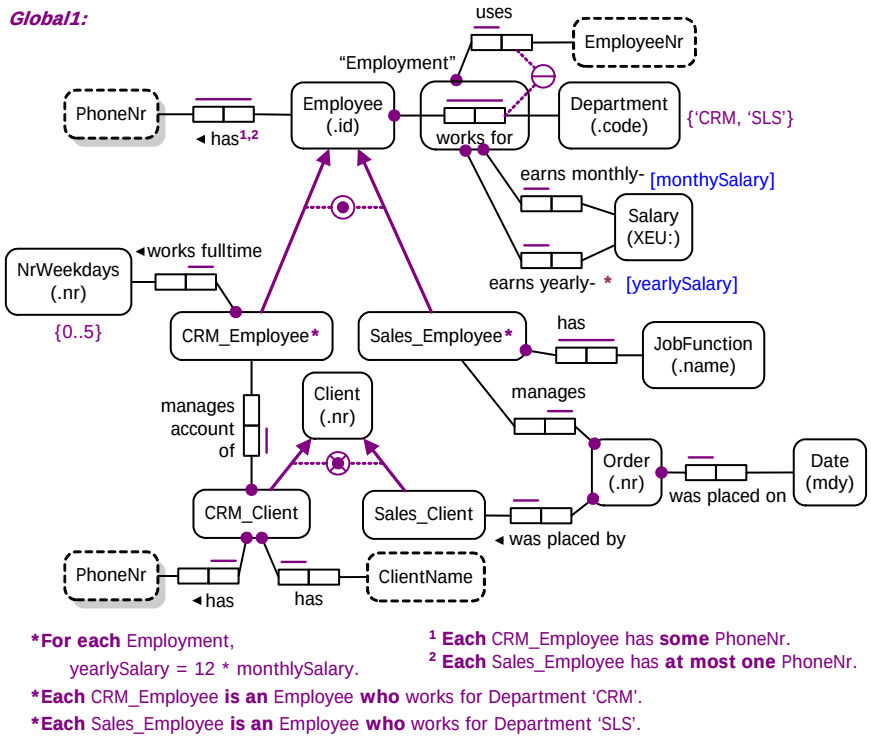

Fig. 2. One way to model the global schema

Assuming the predicates CRM.has_PhoneNr and Sales.has_PhoneNumber are verified by the domain expert to be synonymous, each is mapped to Employee.has_PhoneNr. In this case, the source fact type has different constraints in the local schemas (cf. [13]), so the global fact type is assigned the weakest constraint pattern (optional, *m:n*) while the stronger local constraints are captured by formal textual constraints as shown. Assuming that it is useful to know both monthly and yearly salaries for both CRM and Sales employments, both salary fact types are included as shown, with the yearly salary derived. An alternative is to attach yearly salaries and annual salaries to subtypes CRM_Employee and Sales_Employee respectively. These subtypes are introduced anyway, as each has specific roles as shown.

As discussed earlier (**def**), an alternative to attaching the fulltime weekdays fact type to CRM_Employee is to lift it up to Employee, using a default of 5 placed on the role, assuming the default applies to all employees (if not, this is captured textually). Note that conceptual defaults are still in the process of being added to ORM 2.

ORM 2 supports *derived subtypes and asserted subtypes* [22]. To illustrate these options, the CRM_Employee and Sales_Employee subtypes are derived (indicated by an asterisk and subtype definition), and the CRM_Client and Sales_Client subtypes are merely asserted. We assume that the domain expert has verified that ClientNr does provide a global identifier (unlike EmployeeNr). The rest of the schema should be self-explanatory. This solution is based on applying **syn + diff + conv + sub**.

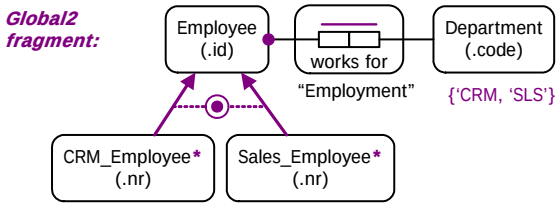

Fig. 3. An alternative way to handle different global and local identification schemes

The Global2 schema fragment in Fig. 3 shows another way to cater for the fact that employees may be identified within their local departments by their local employee number for that department. This notion of *context-dependent reference* was introduced to ORM many years ago [16]. Underlying the subtyping relationships (which are meta-fact types) are implied ground-level fact types that enable instances to be equated (e.g. Employee with EmployeeId '123' = CRM_Employee with CRM_EmployeeNr '246' and Employee with EmployeeId '123' = Sales_Employee with Sales_EmployeeNr '777').

Data extraction has been performed by providing a global schema in which the local data coming from CRM and Sales can be uniformly represented, while data reconciliation has been performed by resolving constraint conflicts by suitable use of differentiation on the global level. Our argumentation is given in terms of an example, but the method we use in moving from local to global can easily be given in more general terms, as described in the following section.

5 Integrating by ψ-maps and Exact Views

Our strategy to integrate a collection of local schemas into a global schema is based on resolving five categories of problems: *homonyms/synonyms*, *default values*, *conversions*, *differentiating roles*, and *subtyping*. Each category has a general solution strategy, as summarized below.

For **hom**, a *LocalObject* (with a role in the local schema) is renamed to *GlobalObject*, in order not to be confused with some Local Object (with a different meaning) having a relation with some other local entity. For **syn**, a *LocalObject1* and *LocalObject2* are renamed to *GlobalObject***.**

For **conv**, a *LocalValue1* and *LocalValue2* are renamed to *GlobalValue*, and the value of *GlobalValue* is derived (alternatively, both are retained and one derived from the other). For **def**, a *LocalEntity1* gets some value of *GlobalObject* as a default value. For **diff** and **sub**, a *LocalEntity* corresponds to subtype *GlobalDiff1* having some fixed default value of type *GlobalValue2*. In *GlobalEntity*, the combination of *GlobalValue1* and *GlobalValue2* is identifying.

In retrospect, we see that we have applied the following principles to resolve integration anomalies.

1. We construct a 1-1 (and sometimes) derived relation between corresponding roles on local and global sides. Such 1-1 relations are called *ψ-maps*.
2. We ensure that all local role values are included in the ψ-map

3. We ensure that all global roles values involved in the ψ-map are included in the objectification

Following 1-3 above, the local role values are injected as corresponding role values. We then proceed by introducing suitable subtypes. The ψ-maps constitute 1-1 relationships between local and global object sets, and in that sense a global object set can be seen as the result of a view (defined by the associated ψ-map) on the local object set. Such views are called exact, because of the 1-1 nature of the ψ-map. Each subtype –on the global level- constructed in such a manner results in an exact view on the corresponding original local entity type (e.g., CRM_Employee, together with its inherited supertype properties, offers an exact view of local entity type CRM.Person).

The CWA and integration

Since the global schema is based on exact views, each local object corresponds to exactly one global object, and each global object corresponds to exactly one combination of local objects. Hence *global understandability* is guaranteed (cf. section 2), entailing that if the local databases all satisfy the Closed World Assumption (CWA), the result also satisfies the CWA. This claim deserves a full formal proof, but due to space limitations such a proof is beyond the scope of this paper.

6 Comparison with UML

The solutions offered in [4, 5] were based on the UML/OCL framework, with heavy use of view-constructs in UML based on the work found in [3]. Views in UML are called *derived classes* (the term was introduced in [8]), and were first given a firm basis in [3] by employing OCL (cf. [35]). Database federations in [4, 5] are defined as certain derived classes (called exact views in [4, 5]) and rely heavily on (sometimes quite lengthy and complicated) OCL-specifications. We think that ORM schemas, as described in this paper, are more readable for defining data federations, since they do not assume the kind of mathematical sophistication necessary to grasp complex OCL-specifications which are often needed to supplement the class diagrams corresponding to aspects of the kind of ORM schemas discussed in this paper, due to the ψ-maps involved and the functional dependencies that have to be preserved in transforming from local to global.

Sustainability

Another advantage of ORM over UML is that its attribute-free style promotes semantic stability. This entails that in an evolutionary setting (where data models can vary in time), an ORM model can be extended without consequences for applications (e.g. queries) that have been defined for earlier versions of the model. Re-modeling in UML, on the other hand, might involve rewriting an attribute to a class in combination with a corresponding association, clearly having consequences for existing appli-

cations on an earlier version of the model. Hence, sustainability of design is an issue here. Database federations are often subject to evolution: local databases may come and go, and flexible adaptive design is an important issue.

7 Conclusion

This paper addressed two major problems (data extraction and reconciliation) in constructing data federations concerning achieving and maintaining consistency and a uniform representation of the data on the global level of the federation. Our approach to constructing a global conceptual schema as the result of integrating a collection of (semantically) heterogeneous component schemas is based on the concept of exact views. We have shown that a global schema constructed in terms of exact views integrates component schemas in such a way that the global schema populates exactly those instances allowed by the local schemas. In this sense, the global schema is equivalent to the set of component schemas from which the global schema is derived.

We have described a modeling framework for data federations based on the Object-Role Modeling (ORM) approach. In particular, we have shown that we can represent exact views within ORM, providing the means to resolve in a combined setting data extraction and reconciliation problems on the global level of the federation.

We note that [5] treats integration of a broader class of constraints than in this paper. In [5] an algorithm has been developed to fully integrate not only structural, but also ad hoc constraints. It is the topic of further research to investigate transfer of these results offered in [5] to the ORM-framework.

References

1. Abiteboul, S. & Douschka, O. 1998, 'Complexity of answering queries using materialized views', *ACM PODS'98*, ACM Press.
2. Abiteboul, S., Hull, R. & Vianu, V. 1995, *Foundations of Databases*, Addison Wesley.
3. Balsters, H. 2003, 'Modeling Database Views with Derived Classes in the UML/OCL-framework', *Proc. UML2003, 6th Int. UML Conf.*, Springer LNCS 2863.
4. Balsters, H. & de Brock, E. 2004, 'An object-oriented framework for reconciliation and extraction in heterogeneous data federations', *Proc. 3rd Int. Conf. Advances in Information Systems*, Springer LNCS 3261.
5. Balsters, H. & de Brock, E. 2004, 'Integration of integrity constraints in federated schemata based on tight constraining', *Proc. OTM 2004*, Springer LNCS 3290.
6. Batini, C., Lenzerini, M. & Navathe, S. 1987, 'A comparative analysis of methodologies for database schema integration', *ACM Computing Surveys*, 18:4, 1987.
7. Bouzeghoub, M. & Lenzerini M. 2001, 'Introduction to: data extraction, cleaning, and reconciliation', *Information Systems* (26), Elsevier Science.
8. Blaha, M. & Premerlani, W. 1998, *Object-oriented Modeling and Design for Database Applications*, Prentice Hall.
9. Cali, A., Calvanese, D., De Giacomo, G. & Lenzerini, M. 2002, 'Data integration under integrity constraints', *Proc. CAISE 2002*, Springer LNCS 2348.
10. Chen, P. P. 1976, 'The entity-relationship model—towards a unified view of data'. *ACM Transactions on Database Systems*, 1(1), pp. 9–36.
11. Demuth, B. & Hussmann, H., 1999, 'Using UML/OCL constraints for relational database design', *Proc. UML'99, 2nd Int. UML Conf.*, Springer LNCS 1723.

12. Demuth, B., Hussmann, H. & Loecher, S. 2001, 'OCL as a specification language for business rules in database applications', *Proc. UML'2001, 4th Int. UML Conf.*, Springer LNCS 2185.
13. Embley, D.W., Xu, L., Ding, Y., 2004, 'Automatic direct and indirect schema mapping: experiences and lessons learned', *SIGMOD Record*, 33(4), pp. 14-19.
14. Golfarelli, M., Lechtenbörger, J., Rizzi, S. & Vossen, G. 2007, 'Schema versioning in data warehouses', *Data and Knowledge Engineering*, in press, Elsevier.
15. Halevy, A. 2001, 'Answering queries using views: A survey', *VLDB Journal* (10).
16. Halpin, T. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
17. Halpin, T. 2004, 'Business Rule Verbalization', *Information Systems Technology and its Applications*, Proc. ISTA-2004, (eds Doroshenko, A., Halpin, T. Liddle, S. & Mayr, H), Salt Lake City, Lec. Notes in Informatics, vol. P-48, pp. 39-52.
18. Halpin, T. 2005, 'Information Modeling in UML and ORM: A Comparison', *Encyclopedia of Information Science and Technology*, vol. 3, ed. M. Khosrow-Pour, Idea Publishing Group, Hershey PA, USA, pp. 1471-5.
19. Halpin, T. 2005. 'Objectification', *Proc. CAiSE'05 Workshops*, vol. 1, eds J. Castro & E. Teniente, FEUP, pp. 519-32.
20. Halpin, T. 2005, 'ORM 2', *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*, eds R. Meersman, Z. Tari, et al., Cyprus. Springer LNCS 3762, pp 676-87.
21. Halpin, T. 2006, 'ORM/NIAM Object-Role Modeling', *Handbook on Information Systems Architectures, 2nd edn*, eds P. Bernus, K. Mertins & G. Schmidt, Springer, Heidelberg, pp. 81-103.
22. Halpin, T. 2007, 'Subtyping Revisited', *Proc. CAiSE'07 Workshops vol. 1,* eds. B. Pernici & J. Gulla, Tapir Academic Press, pp. 131-141.
23. Halpin, T. 2007, 'Fact-Oriented Modeling: Past, Present and Future', *Conceptual Modelling in Information Systems Engineering*, eds. J. Krogstie, A. Opdahl & S. Brinkkemper, Springer, Berlin, pp. 19-38.
24. Hull, R. 1997, 'Managing Semantic Heterogeneity in Databases', *ACM PODS'97*, ACM Press.
25. Lenzerini, M. 2002, 'Data integration: a theoretical perspective', *ACM PODS'02*, ACM Press.
26. Miller, R., Haas, L. & Hernandez, M. 2000, 'Schema mapping as query discovery', *Proc. 26th VLDB Conf.*, Morgan Kaufmann.
27. Mora, S., Trujillo, J. & Song, I. 2006, 'A UML profile for multi-dimensional modeling in data warehouses', *Data and Knowledge Engineering* 59, Elsevier.
28. Object Management Group 2003, *UML 2.0 Superstructure Specification*. Online at: www.omg.org/uml.
29. Object Management Group 2005, *UML OCL 2.0 Specification*. Online at: http://www.omg.org/docs/ptc/05-06-06.pdf.
30. Prat, N., Akoka, J. & Comyn-Wattiau, I. 2007, 'A UML-based data warehouse design method, *Decision Support Systems*, in press.
31. Rahm, E & Bernstein, P. 2001, 'A survey of approaches to automatic schema matching', *VLDB* (10):334-350.
32. Rizzi, S., Abello, A., Lechtenbörger, J. & Trujillo J. 2006, 'Research in Data Warehouse Modeling and Design', *Proc. DOLAP'06*, ACM Press.
33. Sheth, A. & Larson, J. 1990, 'Federated database systems for managing distributed, heterogeneous and autonomous databases', *ACM Computing Surveys* (22).
34. Türker, C. &Saake, G. 2001, 'Global extensional assertions and local integrity constraints in federated schemata', *Information Systems* (25).
35. Warmer, J. & Kleppe, A. 2003, *The Object Constraint Language, 2nd Edition*, Addison-Wesley.