On the Applicability of Schema Integration Techniques to Database Interoperation

Mark W.W. Vermeer and Peter M.G. Apers

Centre for Telematics and Information Technology
University of Twente, Enschede, The Netherlands
{*vermeer,apers*}*@cs.utwente.nl*

Abstract. We discuss the applicability of schema integration techniques developed for tightly-coupled database interoperation to interoperation of databases stemming from different modelling contexts. We illustrate that in such an environment, it is typically quite difficult to infer the real-world semantics of remote classes from their definition in remote databases. However, defining relationships between the real-world semantics of schema elements is essential in existing schema integration techniques. We propose to base database interoperation in such environments on instance-level semantic relationships, to be defined using what we call object comparison rules. Both the local and the remote classifications of the appropriately merged instances are maintained, allowing for the derivation of a global class hierarchy if desired.

1 Introduction

Interoperation among pre-existing, heterogeneous, and autonomous databases has been an important research topic in the last few years. Recently, the trend in database interoperability research is moving towards architectures for interoperation of databases on a scale that goes beyond the context of a single organisation, exploiting the communication facilities offered by world-wide networks. The canonical model used is often an object-oriented one [1]. It has been recognised that such an environment requires flexible and scalable architectures, where users of the component databases are provided with tools to establish importation of information from remote data sources [2,3]. Tightly-coupled approaches, where the schemata of all component databases are unified into a single global schema by a central modelling authority possessing a helicopter view of all component databases [4], are generally agreed to be infeasible in such situations, if only because the component databases may be quite diverse, and no-one can be expected to grasp all information available in the interoperation environment.

Two main approaches towards a more loosely-coupled style of database interoperation can be distinguished. In the multidatabase approach [2], users are expected to define their information needs using a powerful query language with constructs for on-the-fly semantic reconciliation of heterogeneous data. It has been argued, however, that this puts an unacceptable burden on the user, to whom a single logical view of the interoperable databases is no longer presented.

An alternative is the federated approach [5], where a locally integrated schema is composed out of the schema of the local database and the import schema, which is a selection on export schemata of remote databases, by defining relationships between local and imported data. In this paper, we address the question to what extent schema integration techniques developed primarily for tightly-coupled environments are applicable in such environments.

One of the central problems with this approach is that the definition of relationships between local and imported data is far from trivial in a situation where information on the meaning of a remote schema is limited. In tightly-coupled architectures, the schema integrator is supposed to have a helicopter view of all databases, possibly obtained from intensive communication with the DBA's of the participating databases, but in our context a schema integrator must typically thrive on the remote class definitions and some limited additional information. Although many authors have advocated a more loosely-coupled approach to database interoperation, this issue has not been dealt with satisfactorily. Integrated data definition techniques used in federated architectures are usually directly based on schema integration techniques employed in tightly-coupled approaches [6], which in turn are often strongly similar to view integration techniques [7]. All these schema integration techniques require either explicitly or implicitly that (the relationship between) the real-world semantics of the classes to be integrated is known. This is a reasonable assumption in tightly-coupled approaches, but as we will illustrate in this paper, in a federation of databases from multiple modelling contexts this may be surprisingly difficult.

Instance integration [8] has been considered to logically succeed schema integration; i.e. once the relationship between classes defined in different schemata has been determined, the integration of the database instances becomes an issue. More recently, some work has emerged that explicitly considers instances in determining schematic relationships [9,10]. In this paper, we argue that in absence of full knowledge on the semantics of remotely defined classes, instance level semantic relationships form an appropriate basis for database interoperation. In essence, we maintain both the local and the remote classifications on a set of appropriately merged objects. Relationships between local and remote classes may then be derived from relationships between the objects they classify.

The remainder of this paper is organised as follows. In Section 2, we review some basic ideas of schema integration, illustrating that the semantic knowledge required by these techniques may not be available in loosely-coupled environments. In Section 3, we propose an instance-based approach to database integration. In Section 4, we discuss the derivation of integrated objects and classes from instance level relationships. Section 5 then illustrates that several schema-integration techniques are still applicable in our approach. Section 6 presents our conclusions.

2 Schema integration

In this section, we explain why traditional schema integration is not quite suited for federations where knowledge on the semantics of imported schemata is limited. Many techniques for schema integration exist [6]. We do *not* intend to discuss existing schema integration techniques in depth here; rather, we focus on some basic assumptions that do not apply to loosely-coupled federations.

2.1 Classes and entity types

The definition of a class is the result of conceptual modelling performed during database design, in which *entity types* [11] are distinguished. An entity type E is defined as a set of similar real-world objects. Each of the real-world objects grouped in an entity type is described by a set of *properties* associated with the entity type. In the database schema implementing the conceptual model, an entity type is represented by a class C, whose definition contains the properties associated with E. The entity type represented by a class C is called the *Real-World Semantics (RWS)* [12] of C. Moreover, the class may be populated with *database objects* representing some (not typically all) of the real-world objects grouped by the entity type. This set is called the *extension* of C.

As entity types are just sets of real-world objects, set relationships between entity types may exist. In this section, we will use the *subset* relationship as an example. [1] If an entity type E' contains a subset of the real-world objects represented by E, E' is called a *subentity* of E, or E' *isa* E. Let E' be represented in the database by a class C'. Within a database, we expect the following four statements to be equivalent: (1) C' is a subclass of C; (2) the RWS of C' is a subset of the RWS of C (E' *isa* E); (3) the extension of C' is a subset of the extension of C; (4) the set of properties describing C' is a superset of the set of properties describing C.

Example Consider a database DB containing classes **Person** and **Employee**, representing the entity types consisting of the set of persons resp. employees in the real world. Assuming that all employees are persons, Employee is a subentity of Person. Therefore each employee is described by the properties Name and Age, say, in his capacity as a person, and by the additional property Salary, which is used exclusively to describe employees. Moreover, to ensure the integrity of the database, every database object in the extension of **Employee** is contained in the extension of **Person** as well. Hence **Employee isa Person**. Figure 1a illustrates this situation, where C=**Person** and C'=**Employee**. □

2.2 View integration, database integration

Both view integration [7] and (tightly-coupled) database integration are concerned with reconciling multiple conceptual models. The difference is that the

[1] Throughout this section, we will make simplifying assumptions. Our goal here is not to discuss schema integration in depth, but rather to illustrate some features that limit the applicability of these techniques.

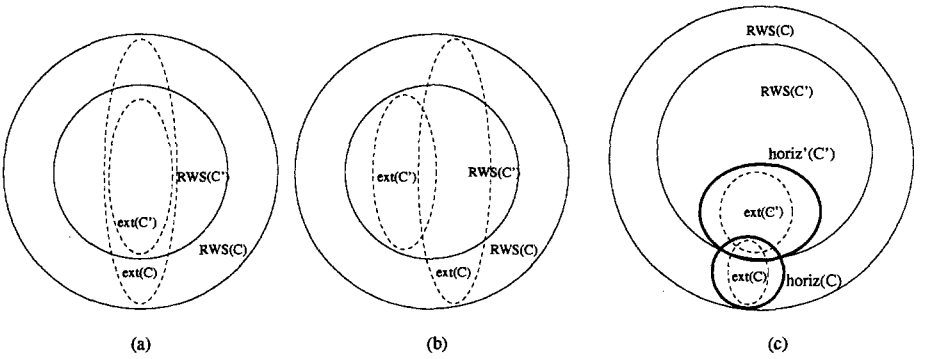

Fig. 1. Real-World Semantics, extensions, and modelling horizons

former is a design activity, whereas the latter deals with schemata that are already populated [6]. Existing work in both of these fields requires some kind of *integration assertions* postulating the relationship between the RWS of the schema elements to be integrated. We here illustrate this for database integration.

Let E be an entity type implemented in a database DB by a class C with properties $prop(C)$ and extension $ext(C)$. Let E' be an entity type implemented in a database DB' by a class C' with properties $prop(C')$ and extension $ext(C')$. Suppose we have the integration assertion $E' \subseteq E$. Note however, that due to modelling autonomy, $prop(C) \subseteq prop(C')$ does not necessarily hold, nor does $ext(C') \subseteq ext(C)$, as illustrated in Figure 1b. An integrated schema would then define virtual classes IC and IC', where $prop(IC') = prop(C') \cup prop(C)$ and $prop(IC) = prop(C)$. Moreover, the extensions of the virtual classes are defined as $ext(IC') = ext(C')$, and $ext(IC) = ext(C) \cup ext(C')$. Now it is assured that IC and IC' satisfy the four aspects of an **isa** relationship listed above.

Note however that whereas this settles things from a conceptual perspective, there are problems providing *values* for the properties of the integration classes. For example, IC'-objects, stemming from $ext(C')$ in DB', do not have values for $prop(C)$, as these properties are defined in the context of DB only. This is a known problem inherent to database integration.

2.3 Database integration in a loosely-coupled environment

So far, we tacitly assumed that set relationships between entity types stemming from different environments were discovered. In the view integration case, this assumption is quite realistic, as different user groups can be asked to provide precise definitions of the entity types they distinguish. The assumption may still be applicable for tightly-coupled database integration performed within a coherent context. However, if database interoperation is performed at such a scale that this kind of communication between database developer and database integrator is infeasible, an integrator is faced with the task of *inferring the entity type a*

class represents from the class definition only. In this subsection, we discuss why this might be surprisingly difficult. We illustrate that, since conceptual modelling is always done within a particular *context* [12], one has to be extremely careful in postulating relationships between entity types from different conceptual models. The main problems we identify are *modelling horizons* and *subjectivity of classification.*

Example: Modelling horizons Suppose database DB is a university database. DB contains a class **Employee**, which is the implementation of an entity type Employee distinguished in its conceptual model. Let DB' be the database of the computer science department of this particular university. DB' contains a class **Person** implementing an entity named Person from its conceptual model. Given the general meaning of these terms, common sense may easily lead an integrator to conclude that **Employee isa Person**. This would be incorrect, however. Even though DB appears to implement the entity 'any employee', we know from the context of DB that this database is concerned with employees working for this particular university only. Moreover, DB' only regards persons as far as they have any connection to the CS-department, such as students and faculty. □

What we encounter here, is that the context in which conceptual modelling is performed introduces a *modelling horizon* to a conceptual model (this corresponds to the way in which the term 'context' is used in [13]). Whenever an entity type E is modelled, it is usually intended to mean 'E as far as it is of any concern to us'. Thus, instead of implementing E, a class C really implements an entity type $horiz(E)$, where $horiz(E) \subseteq E$.

Within a single database, or even within a tightly-coupled environment of databases, this horizon is usually of no importance, as all entity types are implicitly constrained by the *same* horizon. All relationships one expects to hold between entity types are valid. For example,

$E \subseteq E' \Rightarrow horiz(E) \subseteq horiz(E')$

When trying to deduce the entity types implemented by classes stemming from *different* modelling contexts and determining their relationships, however, this modelling horizon is quite relevant indeed, as it may invalidate obvious relationships between entity types. In other words,

$E \subseteq E' \not\Rightarrow horiz(E) \subseteq horiz'(E')$

Example As illustrated in Figure 1c, **Employee** (C') is not a subclass of **Person** (C) when their modelling horizon is taken into account. Even in this simple example, where the contexts of the different databases are quite closely related, deducing the relationships between the entity types represented by the classes from the class definition alone is far from trivial. If an integrator would have knowledge of the contexts of the databases, he would make the correct assertion that these classes have a common virtual superclass **PersonAtOurUniversity**. □

Some authors [14] explicitly disregard modelling horizons by introducing an 'equivalent domain assumption'. In our view, such an approach is infeasible in

loosely-coupled database interoperation, where the modelling horizon can be regarded as an important semantic aspect of a class.

Subjectivity of classification Another important problem with inferring the RWS of remote classes is that conceptual modelling of a real-world domain is essentially a *classification* of 'similar' real-world objects into entity types. The problem with classifications is that they are inherently subjective, which hampers the definition of mappings between such classifications, as in [14,15]. Think of classifications like `FederatedDBS` versus `MultiDBS`, `Hotel` versus `BudgetAccomodation`, or even `AmericanCar` versus `BeautifulThing`. Many real-world examples of different classifications for identical real-world domains exist. It is then often very hard to precisely define the relationships between these classifications.

2.4 Our approach

In this paper, we present a possible alternative for the definition of integration assertions of the form above. In particular, instead of defining semantic relationships between different classifications for a similar real-world domain, we provide for the definition of semantic relationships among the classified *objects*. By applying *both* classifications on the set of appropriately merged objects, relationships between the different classes may then be derived. Note that such comparisons are also made in traditional integration approaches in the phase of *instance integration* [8], which is usually considered to logically succeed schema integration.

3 Instance-based integration

We treat interoperable databases as a collection of database objects, each representing a certain real-world object. Database objects are grouped into classes. Each class is assigned a set of properties by which the objects of that class are described. The set of properties determines the *structure* of an object. Each property has a *domain* from which its values are taken. For *referential* properties, this domain is a class. Each database object provides *values* for its properties. This set of values determines the *state* of a database object. We consider different databases describing a similar real-world domain in different ways.

Example Figure 2 shows a database DB which is used by a university department for purposes of reporting on its output. DB keeps track of publications and master's theses realised by the department. An example database instance is shown. The classification of the database objects reflects DB's purpose. In this context, it is important to distinguish professional publications from scientific ones, and refereed publications from non-refereed ones, as each of these classes of publications have different status.

On the other hand, Figure 3 depicts a database DB' maintained within a certain research project, recording research publications realised within the project. Although the databases have similar application domains, they differ both in the objects distinguished and the classification for these objects, reflecting the different contexts in

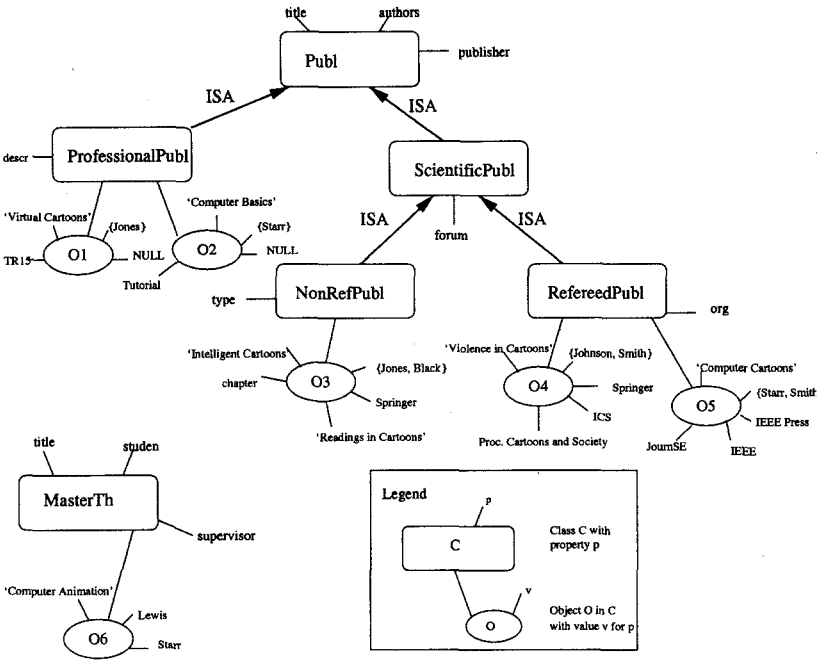

Fig. 2. The department's publications database

which the databases are used. Note that it would be far from easy to a priori define relationships between concepts such as **NonRefPubl** and **Paper**.

In the following, we assume that a user of the department database (the local database) wishes to create an integrated view of his own database and the project's database (the remote database). □

3.1 Object relationships

We now define a number of object relationships, which are basically the instance level equivalent of class relationships distinguished in traditional schema integration [6]. We do not only consider equality of a local object O [2] and a remote object O' [8], but also additional instance relationships, representing semantic relationships that are usually dealt with at the schema level. The following relationships are considered:

- O' may be *equal* to some locally observed object O. In the example, O'_6 is equal to O_4, 'Violence in Cartoons'. The local and remote database thus distinguish the same real-world object.

[2] The term 'object O' is to be interpreted as 'the real-world object represented by database object O' throughout this section. For clarity, a real-world object may also be referred to by the value of a key property.

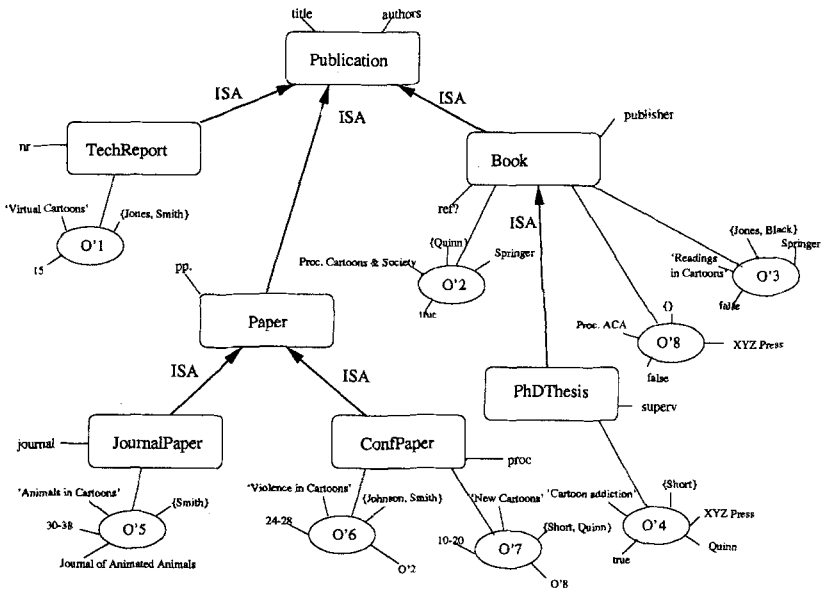

Fig. 3. A project's literature database

- O' may be *similar* to some set of locally observed objects $\{O_i, \ldots O_j\}$, collected in the local class C. We distinguish between strict and approximate similarity.
  Strict similarity occurs when O' logically belongs to C. That is, had O' been observed locally, it would have been classified under C. For example, had the department known about the publication of 'Animals in Cartoons' (O'_5), it would have been classified as a `RefereedPubl` (assuming all journals are refereed).
  Approximate similarity occurs when $C \cup \{O'\}$ is a meaningful class. That is, O' and the objects in C are sufficiently similar to group them into a new, more general class. In the example, a PhD thesis can be seen as approximately similar to a Master's thesis. That is, `MasterTh` $\cup\{O'_4\}$ would represent a meaningful class; let's call it `GradTh`.
- Locally O' would not be seen as an object in its own right, but rather as a set of property values used to describe another object O'', where O'' is equal or similar to a local object O. We say that O is *descriptive* of O''. For example, the object O'_2 is locally seen as a value of the 'forum' property of the object 'Violence in Cartoons'($O'' = O_4$).
- O' would locally be seen as a *constituent* of, or constituted by, an object O'' which is similar or equal to a local object O. That is, O' and O are observed at different levels of granularity. O' is an *aggregation* of O and some other objects, or vice versa. For example, O'_3 ('Readings in Cartoons') is a book containing a chapter by the editors, represented in DB (O_3). Note that we distinguish between aggregation and delegation. The former is used

for composition of objects into a larger object, while the latter expresses relationships among objects.

To express these object relationships [3], we use the following predicates:

1. $Eq(A, B)$ holds iff object A is the same as the object represented by the modelling abstraction B. As illustrated above, B can either be an object (equal objects) or a set of properties (descriptive object).
2. $Sim(A, B[, NewCl])$ holds iff object A is similar to the objects represented by the modelling abstraction B. Again B may be an object or a set of values. If the similarity is approximate, $NewCl$ is the name of the new class classifying A and B.
3. $Aggr(A, B[, Role])$ holds iff the object represented by the modelling abstraction A is an aggregate of the object represented by the modelling abstraction B. A specific role played by B in A may be specified optionally.

Example For the databases of Figure 2 and 3, the relationships sketched above may be postulated as follows:

- Both databases describe technical report no. 15: $Eq(O_1, O'_1)$.
- Proceedings are seen as separate objects in DB' and as values describing conference papers in DB: $Eq(O'_2, O_4.\{\text{forum}\})$ and $Sim(O'_8, \texttt{ConfPaper}.\{\text{forum}\})$.
- Book O'_3 is a value describing O_3 in DB:$Eq(O'_3, O_3.\{\text{forum}\})$.
- We might import Short's PhD thesis as something approximately similar to a master's thesis: $Sim(O'_4, \texttt{MasterTh}, \texttt{GraduationTh})$.
- There are papers appearing in both databases: $Eq(O'_6, O_4)$.
- The chapter by Jones and Black is contained in their book: $Aggr(O_3, O'_3)$.
- A PhD thesis is refereed: $Sim(O'_4, \texttt{RefereedPub})$.
- The journal paper is a refereed one: $Sim(O'_5, \texttt{RefereedPubl})$.
- 'New Cartoons' did not appear in a refereed forum: $Sim(O'_7, \texttt{NonRefPubl})$.
- 'JournSE' is a journal: $Sim(O_5, \texttt{JournalPaper})$.
- By default, we assume that all remote objects are similar to `Publ`.

□

3.2 Object comparison rules

Obviously, when integrating databases, the integrator cannot be expected to inspect individual objects in a pairwise manner to discover relationships between them. We therefore introduce *object comparison rules* as a means to specify conditions under which objects have a certain relationship with one another. For example,

$Sim(O' : \texttt{ConfPaper}, \texttt{RefereedPub}) \leftarrow O'.proc.ref? = true$

$Eq(O' : \texttt{ProfessionalPubl}, O : \texttt{TechReport}) \leftarrow occurs(concat(\text{'TR'}, O.\text{nr}), O'.\text{descr})$

[3] A further relationship between O and O' may be distinguished: a *hidden relationship*, a relationship modelled in neither the local nor the remote database. For example, let O' be the object 'the Netherlands', then O_6 might be written in the Netherlands. Thus, O' might be the value of a new referential property 'written in' of O_6. In this paper, we do not consider such relationships between local and remote objects.

Comparable rules may be defined to specify the other relationships occurring in our example. Observe that such rules are able to deal with so-called *schematic discrepancies* [16], as they describe instances rather than classes. They distinguish themselves from the usual assertions used in schema integration [15] in that the sets of related objects that they define need not coincide with the class extensions of any of the schemata. Note that these rules extend the identity rules of [8] in that they allow for the definition of object relationships other than identity. We discuss these rules further in Section 5.

4 Integrated objects and classes

Having compared local and remote objects, a set of integrated objects and their classification must be determined. The problem that needs to be attacked here can be described as follows:

Given: (1) A set of local objects SL; (2) A classification CL for these objects; (3) A set of remote objects SF; (4) A classification CF for these objects; (5) Relationships between SL and SF;

Find: (1) A new set of objects SI (Subsection 4.1); (2) A classification CI for these objects (Subsection 4.2).

4.1 A new set of objects

Object-value conflicts To arrive at a set of integrated objects, first the local and remote view as to which aspects of the real world are modelled as objects must be reconciled. In particular, whenever an object O in database DB is found to be descriptive of an object O' in DB', i.e. $Eq(O, O'.Props)$ holds, it must be decided whether $Props$ is to be expressed as an object (as in DB) or as a value (as in DB'). Such an *object-value conflict* is usually settled using a fixed strategy, such as 'settle every object-value conflict in favour of the local database', or 'settle every object-value conflict in favour of the object'.

To conform objects and values, *conformed object sets* SLC and SFC are created from SL and SF, respectively. This involves the creation of a *virtual object* from a value-set whenever an object-value conflict is settled in favour of the object, and the hiding of an object whenever an object-value conflict is settled in favour of the value.

Although we cannot treat object-value conflicts in depth here, we do remark that the creation of a virtual object gives rise to new object relationships. Let O'' be a virtual object created from the values of a property set PS of an object O based on the relationship $Eq(O.PS, O')$. This relationship is now replaced by $Eq(O'', O')$. Alternatively, let O''' be a virtual object created from the values of a property set PS of an object O based on the relationship $Eq(O''.PS, O')$, where $O'' \neq O$, but O and O'' both belong to the class C defining the properties PS. Then (approximate) similarity may hold between O''' and the class C' to which O' belongs. Such similarity relationships may be assumed by default or specified through additional rules.

*Example*Consider the relationships $Eq(O'_2, O_4.\{\text{forum}\})$ and $Eq(O'_3, O_3.\{\text{forum}\})$. Note that the strategy of settling all conflicts in favour of the local database would favour the value representation. We here assume that the object preference strategy is applied, however. Thus, virtual local objects O_7 and O_8 are created from the values 'Readings in Cartoons' and 'Proc.Cartoons and Society' of the forum property of O_3 and O_4. Moreover, a virtual object O_9 is created for the forum value 'TrSE' of O_5, as this object belongs to the class `ScientificPubl` in which the property forum is defined. All virtual objects belong to the virtual class `Forum`. The relationships $Eq(O_7, O'_3)$ and $Eq(O_8, O'_2)$ are deduced. □

Integrated objects Given the conformed object sets $SLC = \{O_1, O_2, \ldots, O_n\}$ and $SFC = \{O'_1, O'_2, \ldots, O'_m\}$, we generate an integrated object set $SI = \{O_{ij} | Eq(O_i, O'_j)\} \cup \{O_{i0} | \neg\exists j : Eq(O_i, O'_j)\} \cup \{O_{0j} | \neg\exists i : Eq(O_i, O'_j)\}$. Thus, the integrated object set is a merge of the adapted local and remote object sets, merging duplicate representations of the same real world object into a single integrated representation. Moreover, the aggregation relationship between integrated objects is derived from the aggregation relationship between local and remote objects by substituting the integrated equivalent of each of the related objects.

Example The set SI in our example is formed by

- Representations for real-world objects modelled as objects by both the local and the remote model: O_{11}, O_{46}.
- Foreign objects locally observed only virtually (i.e. as values): O_{73}, O_{82}.
- Virtual local objects not observed in the remote database: O_{90}.
- Objects observed only locally: $O_{20}, O_{30}, O_{50}, O_{60}$.
- Foreign objects not observed locally: $O_{04}, O_{05}, O_{07}, O_{08}$.

Furthermore, O_{73} is at a different level of aggregation than O_{30}. See also Figure 4. □

4.2 A new classification

In essence, we maintain both the local and the remote classification on the integrated object set. That is, initially the set of classes in the integration CI equals $\{\dot{C} | C \in CL\} \cup \{\dot{C'} | C' \in CF\}$, writing $\dot{C}$ to represent the integrated equivalent of the local class C. The set of integrated objects belonging to an integrated class is determined as follows:

- If a local object O_i belongs to a local class C, then the integrated object O_{ij} belongs to $\dot{C}$. (O_i may be virtual, in that case C refers to the corresponding virtual class)
- If a remote object O'_j belongs to a remote class C', then O_{ij} belongs to $\dot{C'}$. (O'_j may be virtual, in that case C' refers to the corresponding virtual class)
- If an object O'_j is strictly similar to a class C, then O_{ij} belongs to $\dot{C}$.
- If an object O'_j is approximately similar to a class C, then a *virtual superclass* CV is introduced. Both O_{ij} and all objects O_{kl} derived from objects in C belong to CV.

– If an integrated object O_{ij} belongs to $\dot{C}$ and $\dot{C}'$, where C and C' stem from different databases, and neither $\dot{C}$ `isa` $\dot{C}'$ nor $\dot{C}'$ `isa` $\dot{C}$ holds, then a *virtual subclass* CV is introduced. O_{ij} belongs to CV.

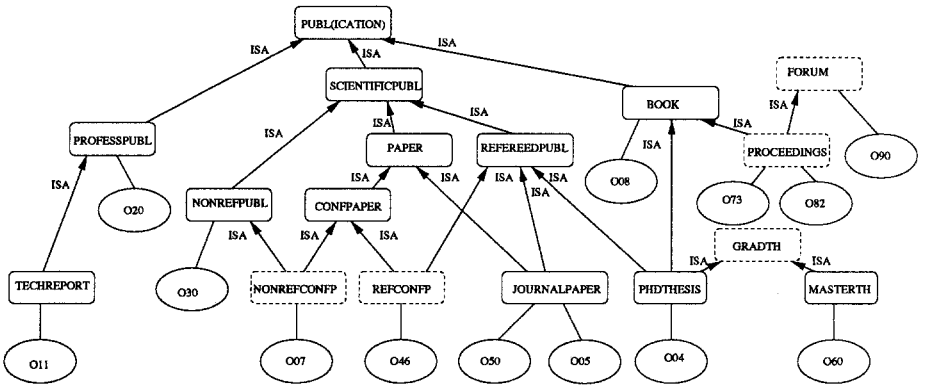

Fig. 4. Integrated objects and classes

Example In our running example, seventeen(!) integrated classes arise. Some examples: `PAPER`=$\{O_{46}, O_{50}, O_{05}, O_{07}\}$, `JOURNALPAPER`=$\{O_{50}, O_{05}\}$, `CONFPAPER`=$\{O_{46}, O_{07}\}$, the local virtual class `FORUM`=$\{O_{73}, O_{82}, O_{90}\}$, and integrated virtual classes such as `REFCONFP`=$\{O_{46}\}$ and `PROCEEDINGS`=$\{O_{73}, O_{82}\}$. The complete set of integrated classes is depicted in Figure 4. □

Although in principle the definition of integrated extensions for local classes is an appropriate basis for loosely-coupled database interoperation, in some situations it is desirable to derive an integrated class hierarchy describing the relationships between local and remote classes. Such class relationships follow from relationships between the integrated extensions. That is, the integrated class hierarchy is a consequence of relationships between local and remote objects rather than being defined explicitly. In particular, the following relationships between the integrated equivalent of a local class $\dot{C}$ and the integrated equivalent of a remote class $\dot{C}'$ may arise:

1. $\dot{C}$ `isa` $\dot{C}'$, i.e. $\forall O \in C \exists O' \in C' \mid Eq(O, O') \vee Sim(O, C')$ (or $\dot{C}'$ `isa` $\dot{C}$ etc.).
2. $\dot{C}$ and $\dot{C}'$ are identical, i.e. $\dot{C}$ `isa` $\dot{C}'$ and $\dot{C}'$ `isa` $\dot{C}$.
3. $\dot{C}$ and $\dot{C}'$ have a common virtual subclass CV, i.e. $\exists O \in C, O' \in C' \mid Eq(O, O') \vee Sim(O, C') \vee Sim(O', C)$.
4. $\dot{C}$ and $\dot{C}'$ have a common virtual superclass CV, i.e. $\forall O \in C, O' \in C' \mid Sim(O, C', CV) \vee Sim(O', C, CV)$.
5. $\dot{C}$ and $\dot{C}'$ have CV-generalisable elements, i.e. $\exists O' \in C' \mid Sim(O', C, CV)$ (or $\exists O \in C$ etc.). Here CV is a superclass of $\dot{C}$ and a virtual subclass CV' of $\dot{C}'$, where $CV' = \{O' | O' \in C' \wedge Sim(O', C, CV)\}$.

The class hierarchy for our example is depicted in Figure 4.

5 Specifying instance-based interoperability

5.1 Description overlap

Having determined a set of integrated objects and a corresponding classification, we turn to the discussion of structure and state of integrated objects. We touch upon this subject only briefly, as it has been treated exhaustively by existing schema integration methodologies.

In principle, we could simply define the structure and state of integrated objects as the merge of their local and remote counterparts. However, there is the possibility of *description overlap*; i.e. the local and the remote database contain descriptions of identical real-world properties. Description overlap may occur in any of the kinds of object relationships we distinguished. To establish the description overlap between objects stemming from classes C and C', *property equivalence assertions* between their properties must be defined. A property equivalence assertion states that a local and a remote property represent the same real-world property. In general, m properties of C may be equivalent to n properties of C'. That is, a property p of C derived from these m properties may be equivalent to a similar derived property p' of C'. Equivalent local and remote properties are merged into a single integrated property. Equivalent properties need not have identical *domains*, however. For integration purposes, a *conversion function* [17] between these domains must be defined. Even when domains of equivalent properties have been mapped to one another, different databases may disagree on the value chosen from that domain to represent a property p of a certain object O. Some *decision function* [18] is then applied.

Some additional observations can be made for *referential properties*. As the domain of a referential property is a class, conversion and integration of equivalent referential properties is influenced by the integrated class hierarchy. Conversion of a referential property is implicit; the integrated property refers to integrated objects instead of local objects. To infer an integrated delegation hierarchy, we again use an instance-based approach. That is, the domain of an integrated referential property is the most specific integrated class containing all referenced integrated objects.

Example Consider the integrated class `REFCONFP`. Local and remote objects merged in this class have the equivalent referential properties 'proc' resp. 'forum'. The domain class `PROCEEDINGS` is now inferred for the integrated referential property. □

5.2 Specification of database interoperation

A database interoperation specification consists of object comparison rules accompanied by property equivalence assertions. To express object relationship conditions, a tailored equality predicate $=_{prop}$ is used to compare values of equivalent properties, which is defined modulo any conversion function defined on the properties involved.

Object relationships and inheritance Let C be a local class and C'_1, C'_2 be remote classes such that C'_2 `isa` C'_1. Then

$\forall O, O' \mid Eq(O : C, O' : C'_2) \Rightarrow Eq(O : C, O' : C'_1)$ and
$\forall O, O' \mid O' \in C'_2 \wedge Eq(O : C, O' : C'_1) \Rightarrow Eq(O : C, O' : C'_2)$ and
$\forall O \mid Sim(O : C, C'_2) \Rightarrow Sim(O : C, C'_1)$

These laws can be used to derive additional object relationships from those defined directly by the object comparison rules. Equality is a stronger notion than similarity; hence equality rules are evaluated before similarity rules. A specifier may exploit these laws by defining equality rules on as general classes as possible, but similarity is defined on the most specific class.

The following requirements are made to ensure that Eq has the intended semantics. (1) Every equality rule defines a one-to-one mapping; and (2) the definition of equality rules of the form $Eq(O : C, O' : C') \leftarrow \psi$ when there also exists a rule $Eq(O : C_{sup}, O' : C'_{sup}) \leftarrow \phi$ where C `isa` C_{sup} or C' `isa` C'_{sup}, is allowed only if $\psi \not\Rightarrow \phi$. Furthermore, to avoid inconsistencies, the definition of similarity rules of the form $Sim(O : C, C') \leftarrow \psi$ is not allowed if there exists a rule $Sim(O : C, C'_{sup}) \leftarrow \phi$ where C' `isa` C'_{sup} and $\psi \Rightarrow \neg\phi$.

6 Discussion

The purpose of this paper has been to demonstrate the use of extensions in loosely-coupled database interoperation. We have argued that instance level relationship specifications can be the basis of a database interoperation mechanism. We believe that the information provided by the database extension can compensate for the decrease in knowledge of remote schemata and their modelling contexts, which is inherent to loose coupling. Thus, we avoid having to define class relationships, which we believe to be error-prone in view of modelling horizons and subjectivity of classification.

It may be observed that the use of object comparison rules is similar to works exploiting a query language for defining multidatabase mappings such as [19], and also [20]. Distinguishing features are the various object relationships we defined, leading to a style of specification which is suitable to loosely-coupled database interoperation, and the possibility of deriving a global class hierarchy. Note that in mediator systems aimed at interoperation of data not necessarily managed by a DBMS, instance-based approaches have been developed as well [21,22]. Compared to these contexts the schema information available in a federation of databases allows for the derivation of a schema for the integrated instances obtained through the interoperation mechanism.

Since we derive an integrated class hierarchy based on instance-level information, our approach is primarily targeted at relatively stable environments where discrepancies exist between the viewpoints of different databases containing related data. As in principle changes in the local extensions may result in changes to this integrated class hierarchy, our approach must be extended to also deal with dynamic environments where extensional relationships between classes are unstable. For example, a query mechanism capable of dealing with schema evolution at the integrated level could be devised. Alternatively, object comparison

rules could be used in the context of import/export-based database interoperation [23], where a global hierarchy is not used at all. This is a subject of our current research.

We are also working on the inclusion of methods and constraints in our integration strategy. In particular, we are interested in the additional semantics that local methods and constraints provide for the local data structures, and the way in which this information can be used to detect inconsistencies in interoperation specifications. Some results in this direction can be found in [24].

References

[1] E. Pitoura, O. Bukhres and A. Elmagarmid, "Object orientation in multidatabase systems," *Comput. Surv.*, 27, no. 2, pp. 141–195, June 1995.

[2] W. Litwin, L. Mark and N. Roussopoulos, "Interoperability of multiple autonomous databases," *ACM Computing Surveys*, 22, no. 3, pp. 267–293, Sept. 1990.

[3] G. Wiederhold, "Value-added mediation in large-scale information systems," in *IFIP Data Semantics (DS-6), Atlanta, Georgia.* 1995.

[4] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous and autonomous databases," *ACM Computing Surveys*, 22, no. 3, pp. 183–236, Sept. 1990.

[5] D. Heimbigner and D. McLeod, "A federated architecture for information management," *ACM Trans. Off. Inf. Syst.*, 3, no. 3, pp. 253–278, July 1985.

[6] C. Batini, M. Lenzerini and S. B. Navathe, "A comparative analysis of methodologies for database schema integration," *ACM Computing Surveys*, 18, no. 4, Dec. 1986.

[7] S. Spaccapietra and C. Parent, "View integration: A step forward in solving structural conflicts," *IEEE Trans. Knowl. & Data Eng.*, 6, no. 2, pp. 258–274, Apr. 1994.

[8] E-P. Lim, J. Srivastava, S. Prabhakar and J. Richardson, "Entity identification in database integration," in *Proceedings Ninth International Conference on Data Engineering, Vienna, Austria, Apr. 19–23, 1993.* Washington, DC: IEEE Computer Society Press, pp. 294–301, 1993.

[9] W-S. Li and C. Clifton, "Semantic integration in heterogeneous databases using neural networks," in *Proceedings of Twentieth International Conference on Very Large Data Bases, Santiago, Chile, Sept. 12–15, 1994*, J. Bocca, M. Jarke and C. Zaniolo, Eds. San Mateo, CA: Morgan Kaufmann Publishers, pp. 1–12, 1994.

[10] M. Garcia-Solaco, F. Saltor and M. Castellanos, "A structure based schema integration methodology," in *Proceedings Eleventh International Conference on Data Engineering, Taipei, Taiwan, Mar. 6–10, 1995.* Washington, DC: IEEE Computer Society Press, pp. 505–512, 1995.

[11] P. P. Chen, "The entity-relationship model - Towards a unified view of data," *ACM Trans. Database Syst.*, 1, no. 1, pp. 9–36, 1976.

[12] A. Sheth and V. Kashyap, "So far (schematically) yet so near (semantically)," in *IFIP Interoperable Database Systems (DS-5), Lorne, Victoria, Australia, 16–20 November, 1992.* Amsterdam: North-Holland, pp. 283–312, 1993.

[13] M. Garcia-Solaco, F. Saltor and M. Castellanos, "Semantic heterogeneity in multidatabase systems," in *Object-oriented multidatabase systems*, O. A. Bukhres and A. K. Elmagarmid, Eds. Englewood Cliffs, NJ: Prentice-Hall, pp. 129–195, 1996.

[14] M. V. Mannino, S. B. Navathe and W. Effelsberg, "A rule-based approach for merging generalization hierarchies," *Inf. Syst.*, 13, no. 3, pp. 257–272, 1988.

[15] Y. Dupont, "Resolving fragmentation conflicts in schema integration," in *13th International Conference on Entity-Relationship Approach.* New York–Heidelberg–Berlin: Springer-Verlag, pp. 513–532, 1994.

[16] R. Krishnamurthy, W. Litwin and W. Kent, "Interoperability of heterogeneous databases with schematic discrepancies," in *Proc. First Intl. Workshop on Interoperability in Multidatabase Systems.* Montvale, NJ: IEEE Press, pp. 144–151, 1991.

[17] W. Kent, "Solving domain mismatch and schema mismatch problems with an object-oriented database programming language," in *Proceedings of Seventeenth International Conference on Very Large Data Bases, Barcelona, Spain, Sept. 3–6, 1991*, G. M. Lohman, A. Sernadas and R. Camps, Eds. San Mateo, CA: Morgan Kaufmann Publishers, pp. 147–160, 1991.

[18] U. Dayal and H-Y. Hwang, "View definition and generalization for database integration in a multidatabase system," *IEEE Trans. Software Eng.*, 10, no. 6, pp. 628–645, Nov. 1984.

[19] E. Kuehn and T. Ludwig, "VIP-MDBS: A logic multidatabase system," in *Proceedings of International Symposium on Databases in Paralleland Distributed Systems, Austin, Texas, Dec. 5–7, 1988* , S. Jajodia, W. Kim and A. Silberschatz, Eds. Montvale, NJ: IEEE Press, pp. 190–201, 1988.

[20] M. H. Scholl, H-J. Schek and M. Tresch, "Object algebra and views for multi-objectbases," in *Distributed object management*, M. T. Oszu, U. Dayal and P. Valduriez, Eds. San Mateo, CA: Morgan Kaufmann Publishers, pp. 353–374, 1994.

[21] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman and J. Widom, "The TSIMMIS approach to mediation: Data models and languages," Stanford University, Stanford, CA, 1995.

[22] V. S. Subrahmanian, S. Aldali, A. Brink, R. Emery, J. J. Lu, A. Rajput, T. Rogers, R. Ross and C. Ward, "HERMES: A heterogeneous reasoning and mediator system," University of Maryland, Maryland, 1995.

[23] D. Fang, S. Ghandeharizadeh, D. McLeod and A. Si, "The design, implementation, and evaluation of an object-based sharing mechanism for federated database systems," in *Proceedings Ninth International Conference on Data Engineering, Vienna, Austria, Apr. 19–23, 1993.* Washington, DC: IEEE Computer Society Press, pp. 467–475, 1993.

[24] M. W. W. Vermeer and P. M. G. Apers, "The role of integrity constraints in database interoperation," in *Proceedings 22nd International Conference on Very Large Databases (VLDB'96), Bombay, India.* San Mateo, CA: Morgan Kaufmann Publishers, 1996.