

# The Role of Integrity Constraints in 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 present an approach to database interoperation that exploits the semantic information provided by integrity constraints defined on the component databases. We identify two roles of integrity constraints in database interoperation. First, a set of integrity constraints describing valid states of the integrated view can be derived from the constraints defined on the underlying databases. Moreover, local integrity constraints can be used as a semantic check on the validity of the specification of the integrated view. We illustrate our ideas in the context of an instance-based database interoperation paradigm, where objects rather than classes are the unit of integration. We introduce the notions of *objectivity* and *subjectivity* as an indication of whether a constraint is valid beyond the context of a specific database, and demonstrate the impact of these notions.

## 1 Introduction

Although the interoperation of autonomous legacy databases has been an important research subject for the last few years, so far little attention has been paid to the role of database integrity constraints in this

context. Existing research has focused on the definition of an integrated view on the schemata of component databases, either in a loosely-coupled or a tightly-coupled manner, and the possibilities for querying and updating such a view [ShL90]. Considering the importance of integrity constraints for the semantics of a database, and the attention that the issue of *semantic heterogeneity* [DKM93] among interoperable databases has received, the importance of our subject is evident, however. At least two important questions related to integrity constraints in a database interoperation environment arise almost immediately.

The first of these questions is: *Given the integrity constraints defined on the component databases, what integrity constraints can be defined on the integrated view of these databases?* This is the subject of the scarce literature on constraints in database interoperation known to us [AQF95, RPG95]. Global integrity constraints thus obtained could for example be used in optimising queries against the integrated view, eliminating subqueries which are known to yield empty results. Another possible use is in the validation of update transactions, preventing the formulation of subtransactions which will certainly be rejected by the local transaction manager. We feel that existing work falls short of treating this question exhaustively. To illustrate this claim, we here present an overly simplified example introducing two observations not seen in present literature.

### Example

Consider two personnel databases  $DB_1$  and  $DB_2$  held by different departments of a certain company. Both contain a class *Employee* with attributes (ssn, salary, trav-reimb).  $DB_1$  has two constraints defined on employees: (1)  $\text{trav-reimb} \in \{10, 20\}$ , and (2)  $\text{salary} < 1500$ . In  $DB_2$ , a single constraint is defined: (1)  $\text{trav-reimb} \in \{14, 24\}$ . If  $DB_1$  and  $DB_2$  are integrated into a view  $DB_{int}$ , what are the constraints to be defined

---

*Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.*

Proceedings of the 22nd VLDB Conference  
Mumbai (Bombay), India, 1996

on employees in  $DB_{int}$ ?

Our first observation is that constraint (2) of  $DB_1$  is not necessarily a valid constraint for  $DB_{int}$ . That is, this constraint may represent a business rule adhered to by a specific department rather than being an axiom for employees in general. This is what we call a *subjective constraint*; it is valid in the context of a specific database only.

With regard to constraint (1) of both  $DB_1$  and  $DB_2$ , expressing possible daily travel reimbursement tariffs, we observe the following. Suppose this company has multi-department projects; i.e. employees may be registered by more than one department. At first sight, there appears to be a conflict between these constraints. However, suppose that the company has the following policy with respect to business trips: Trips made on behalf of multiple departments are reimbursed based on the average of the tariffs of the departments involved. Thus, the value of the trav-reimb attribute of such an employee in  $DB_{int}$  is defined as the average of the local values. Note that for such employees, we can *derive* a global constraint (1) trav-reimb  $\in \{12, 17, 22\}$ ; the apparent conflict has been solved by the way the global values are defined. Apparently, there is a relationship between the determination of global property values and global constraints. In this paper, we will discuss the implications of property value decision functions on global constraints in more detail.  $\square$

We also identify a second question: *Is a given specification of an integrated view on a number of component databases consistent with the integrity constraints defined on those databases, and if not, how can this specification be corrected?* In particular, the set of global constraints derived from local constraints and a given integration specification may be inconsistent. We discuss options for solving such an inconsistency.

In our treatment of integrity constraints, we distinguish among several kinds of relationships that *objects* from different databases may have. For example, in the context discussed above, there is a sharp distinction between an employee  $e$  occurring in  $DB_1$  only, and an employee  $e'$  occurring in *both* databases, as far as constraints are concerned. The global state of  $e$  is entirely determined from  $DB_1$ , and so are the constraints valid on  $e$ . The determination of the global state of  $e'$ , however, involves both  $DB_1$  and  $DB_2$ , the consequences of which we discussed above.

We therefore illustrate our ideas in the context of our approach to database importation presented in [VeA96], in which objects rather than classes are the unit of integration. A summary of this approach is provided in Section 2. In Section 3, we discuss the relationship between integrity constraints and our integra-

tion paradigm. Section 4 is concerned with expressing local and foreign constraints in a compatible way, so that they can be compared in Section 5, concentrating on the two questions identified above. Section 6 presents our conclusions. In our opinion, our findings are sufficiently general to be valid beyond the specific framework used here. Our discussion serves to illustrate our conviction that 'legacy constraints' should be dealt with by any database interoperation methodology. engineering of an existing relational database and its applications

Note that in contrast to e.g. [CGW96,GrW96], we do not deal with the formulation and enforcement of global integrity constraints which inherently concern multiple component databases (so-called distributed integrity constraints). The scope of this paper is restricted to constraints that are being enforced by the component databases, and their implications on the global view.

## 2 Structural integration

In this section, we summarise our instance-based approach to database interoperation, which will form the context for our discussion of integrity constraints. This approach is characterised by the adaptation of existing schema integration techniques [BLN86] to be applicable at the instance level of database interoperation, thus avoiding the explicit mapping of the different classifications used by the different component databases to describe a similar application domain.

We furthermore introduce an example that will be used throughout this paper. The example focuses on constraints; it is not intended to illustrate the features of our integration methodology in full. For a more complete discussion, we refer to [VeA96].

In examples throughout this paper, we will use the syntax of the object-oriented specification language TM [BBZ93], which allows for the expression of first-order constraints on an object-oriented database. The syntax is rather self-explanatory. Semantically rich specifications such as those expressible in TM are not always available for existing databases. Typically, such specifications are obtained through reverse engineering, as discussed in [VeA95].

As to the constraints introduced in our example of Figure 1, we note the following. In our discussion, we distinguish among *object constraints*, *class constraints*, and *database constraints* that may be defined in each of the local models, depending on whether they constrain the state of a single (complex) object; a set of objects from a single class, or a set of objects from different classes. Object constraints should be read as implicitly universally quantified over all instances of a class. TM supports these constraint types at the

specification level, and design tools supporting proper classification of constraints exist [FKS94]. Note that we consider static constraints only.

## 2.1 Instance-based database interoperation

In [VeA96] we outlined an instance-based approach to database interoperation. As argued in length there, we consider objects rather than classes to be an appropriate basic unit of integration. Such a view is also often taken in mediator systems aimed at interoperation of data not necessarily managed by a DBMS (e.g. [GPQ95]). In short, the motivation for our approach is the argument that in absence of a common semantical context, it is more feasible for disparate sources to agree on relationships among the specific real-world objects that they describe, than to agree on the semantics of possible classifications for those objects.

Consider the two databases of which a TM-specification is given in Figure 1 (some TM-details have been left out for presentation purposes). Database *CSLibrary* is kept by the library of a computer science department to record its collection of books. It is also used as an internal reference judging the importance of publication fora. Database *Bookseller* is a database kept by a scientific bookseller through whom the library occasionally acquires new literature. The bookseller has an importance rating system comparable to the one maintained by the library. From the specifications it can be seen that the different contexts in which the databases are used lead to different classifications for a similar application domain. For example, the library distinguishes scientific publications from professional ones, whereas the bookseller distinguishes proceedings from monographs. We assume that local users of *CSLibrary* wish to define a virtual integrated view of their local database extended with data imported from *Bookseller*.

Traditional schema integration techniques [BLN86] would require the definition of the relationships between the real world semantics of concepts defined in different databases. We argue that in absence of a common semantical context, this is typically extremely difficult. For example, what precisely does the bookseller mean by a *Monograph*? Would we always call it a *ScientificPubl* or possibly a *ProfessionalPubl*? In general, schema integration techniques require the designers of the different databases to reach an agreement on this and similar questions, something which is typically not possible in database interoperation.

Therefore, we argue that in such a context, objects rather than classes are the appropriate unit of integration. We require the definition of relationships between *objects*, by specifying conditions under

which objects from different classes are related in a certain way. In our example, we would define that *Monograph* and *ScientificPubl*-objects carrying the same *isb-number* (modulo representation differences, not discussed here) are in fact identical objects, and that any *Proceedings* object with *ref?=true* would be classified as *RefereedPubl* by the library. We maintain the classification of both databases on the set of appropriately merged objects. It might then *appear* from their integrated extensions that, for example, the class *Monograph* is a subclass of the *ScientificPubl*-concept, but this is a *result* of object relationships rather than being defined by a designer or enforced by the integration mechanism.

## 2.2 Specifying database interoperability

Our approach requires a designer to specify conditions under which a certain relationship  $\rho$  between a remote object  $O'$  and a local object  $O$  or class  $C$  holds<sup>1</sup>. The relationships we distinguish are:

- **Equality.**  $O$  and  $O'$  represent the same real world object [LSP93]. This is represented as  $Eq(O', O)$ .
- **Strict similarity.**  $O'$  would locally be classified under  $C$ . This is represented as  $Sim(O', C)$ .
- **Approximate similarity.** Locally  $C \cup \{O'\}$  can be regarded as a more general virtual class  $C''$ . This is represented as  $Sim(O', C, C'')$ .
- **Descriptivity.** Locally  $O'$  is considered a set of values  $S$  describing an object  $O''$  which is identical to a local object  $O$  or similar to a local class  $C$ . This is represented as  $Eq(O', O.S)$  or  $Sim(O', C.S)$ .

In [VeA96] we also defined the constituency relationship, but this is not relevant here. We require the specification of *object comparison rules* of the form  $\rho \leftarrow \Psi$ , where  $\rho$  is any of the relationships listed above, and  $\Psi$  is a conjunction of first-order logic predicates, which might involve additional information such as correspondence tables etc.

Moreover, *property equivalence assertions* must be formulated, specifying to what extent the descriptions provided by  $DB$  and  $DB'$  overlap. These assertions are of the form  $propeq(C.p, C'.p', cf, cf', df)$ , where:

- $p, p'$  are basic or derived local and remote properties, respectively,

<sup>1</sup>In the remainder of this paper, we use the conventions for symbols  $s$  to refer to the local database,  $s'$  to refer to the remote database, and  $\dot{s}$  to refer to the integrated view of these databases.

Database CSLibrary

Class Publication

attributes title : string  
isbn : string  
publisher : string  
shopprice : real  
ourprice : real

object constraints

ocl: ourprice <= shopprice

oc2: publisher in KNOWNPUBLISHERS

class constraints

ccl: key isbn

cc2: (sum (collect x for x in self) over ourprice) < MAX

end Publication

Class ScientificPubl isa Publication

attributes editors : Pstring  
rating : 1..5

class constraints

ccl: (avg (collect x for x in self) over rating) < 4

end ScientificPubl

Class RefereedPubl isa ScientificPubl

attributes avgAccRate : real

object constraints

ocl: rating >= 2

end RefereedPubl

Class NonRefereedPubl isa ScientificPubl

attributes authAffil : string

object constraints

ocl: rating <= 3

end NonRefereedPubl

Class ProfessionalPubl isa Publication

attributes authors : Pstring

end ProfessionalPubl

Database Bookseller

Class Item

attributes title : string  
isbn : string  
publisher : Publisher  
authors : Pstring  
shopprice : real  
libprice : real

object constraints

ocl: libprice <= shopprice

class constraints

ccl: key isbn

end Item

Class Proceedings isa Item

attributes ref? : boolean  
rating : 1..10

object constraints

ocl: publisher.name='IEEE' implies ref?=true

oc2: ref?=true implies rating >= 7

oc3: publisher.name='ACM' implies rating >= 6

end Proceedings

Class Monograph isa Item

attributes subjects : Pstring

end Monograph

Class Publisher

attributes name : string  
location : string

end Publisher

Database constraints

db1: forall p in Publisher exists i in Item |  
i.publisher = p

Figure 1: Example databases with constraint definitions

- $cf, cf'$  are *conversion functions* mapping the domains of  $p$  and  $p'$  to a common domain  $D$ , and value.

- $df : D \times D \rightarrow D$  is a *decision function* which determines a global value for the property given possibly different local and remote values. We require that for each decision function  $df$ ,  $\forall a \in D | df(a, a) = a$ . In our view, functions such as *sum* used e.g. in [DaH84] define derived global properties rather than determining values for equivalent local and remote properties.

### Example

For our example databases, we list some sample object comparison rules. We also list some property equivalences, omitting obvious ones. We use predefined conversion functions such as *id*, the identity function, and decision functions such as *trust*, which assigns a specific database as the primary source for a property's

```
Eq(O:Publication, O':Item) ← O.isbn=O'.isbn
Eq(O:Publication.{publisher}, O':Publisher)
  ← O.publisher=O'.name
Sim(O':Proceedings,RefereedPubl)← O'.ref?=true
Sim(O':Proceedings,NonRefereedPubl)← O'.ref?=false
Sim(O:ScientificPubl,Proceedings)
  ← contains(O.title, 'Proceed')
```

```
propeq(Publication.ourprice, Item.libprice,
  id, id, trust(CSLibrary))
propeq(Publication.shopprice, Item.shopprice,
  id, id, trust(Bookseller))
propeq(Publication.publisher, Publisher.name, id, id, any)
propeq(ScientificPubl.rating, Proceedings.rating,
  multiply(2), id, avg)
propeq(ScientificPubl.editors, Item.authors, id, id, union)
```

□

### 2.3 The integrated view

As a result of a specification as defined above, an integrated or global view of the local and the remote database can be constructed. This construction is a two-step process of *conformation* and *merging* analogous to the steps distinguished for schema integration in [BLN86]. Our discussion here is necessarily brief; the interested reader is referred to [VeA96].

In the conformation step, the local and remote database are brought into a common semantical context, so that they can be merged. This involves the settling of object-value conflicts resulting from descriptivity relations between objects. This is done by creating virtual objects from values and/or casting objects into property values describing other objects. In the resulting *conforming object sets* *SLC* and *SFC*, the local and remote use of objects versus values has been conformed. In our example, the description of a publisher as a value describing a publication or as a separate object must be conformed. If for example the object view is taken, virtual *VirtPublisher*-objects are created from the values of *Publication.publisher*.

Equivalent local and remote properties  $p$  and  $p'$  are turned into *conforming properties*  $p_c$  and  $p'_c$  by assigning them identical names and converting them to identical domains. If virtual objects have been derived from objects the property was defined on originally, the conforming properties may be assigned to these virtual objects. Examples include the renaming of 'ourprice' to 'libprice', the conversion of library and bookseller's publication ratings to a common scale, and the assignment of the 'publisher' attribute values to a new attribute 'name' of the virtual publisher objects.

In the merging step, objects from *SLC* and *SFC* between which an equivalence relationship has been determined, are merged into a single global object. Equivalent properties are merged into an integrated property and assigned to the integrated class hierarchy. Moreover, the value of global properties is determined from the conformed local and remote ones, using a decision function where applicable.

The crux of applying these well-known techniques directly at the instance level is that a classification for the integrated view is now formed by applying *both* the local and the remote classification to the global object set. As the global object set is a merge of the local and remote one, relationships between local and remote classes may thus be detected; for example,  $C$  is a  $C'$  iff  $\forall O \in C \exists O' \in C' : Eq(O, O') \vee Sim(O, C')$ . Thus, the global class hierarchy is a result of object relationships rather than being defined explicitly.

As an example, if it turns out that some, but not all, of the objects in *Proceedings* and

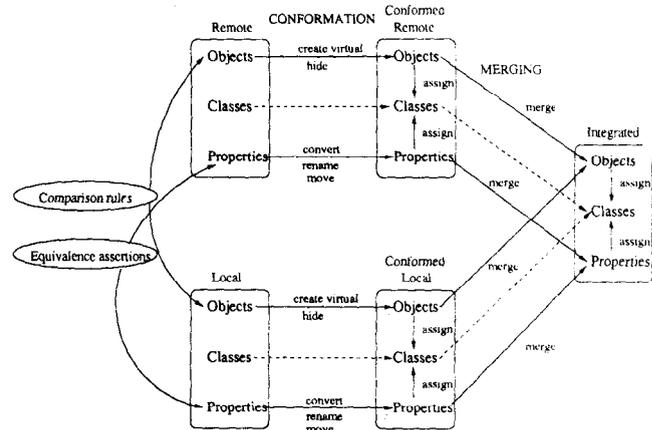


Figure 2: Instance-based database interoperation. *RefereedPubl* are similar, a virtual global subclass *RefereedProceedings* containing these objects arises, which is a subclass of both *Proceedings* and *RefereedPubl*. The process of conformation and merging is illustrated in Figure 2.

### 3 Object comparison rules and constraints

Recall that one of the questions posed in the introduction was concerned with detecting possible inconsistencies between constraints and the specification of the integration. We must distinguish between two cases here: (1) The specification of the integration is in conflict with the *local* constraints; (2) The specification of the integration leads to conflicts among *global* constraints that are a consequence of this specification. We do not deal with the second of these cases until we have discussed the derivation of such global constraints in Section 5. This section briefly considers the first case.

The condition part of object comparison rules, like object constraints, imposes conditions to be satisfied by object instances. We here distinguish between two types of object comparison rule conditions:

- *Interobject conditions* These are conditions such as  $O.isbn = O'.isbn$ , defining conditions involving both objects to be compared.
- *Intraobject conditions* These are conditions on one of the objects to be compared only, such as  $O.ref? = true$ . Thus, intraobject conditions are conditions that local (or remote) object must satisfy to be a candidate for having this relationship in the first place.

The strong resemblance between intraobject conditions and object constraints has two consequences. First, the intraobject conditions on objects of a class  $C$  imposed by a rule  $r$  must not conflict with the object

constraints defined by  $C$ . This is a conflict as alluded to in the introduction. Second, from the object constraints of  $C$  and the intraobject conditions defined by  $r$ , *implied object constraints* may be derived, which are subsequently treated like regular object constraints in the integration process.

#### Example

Consider the object comparison rule  $Sim(O':Proceedings,RefereedPubl) \leftarrow O'.ref? = true$ . The condition defined in this rule is an intraobject condition which is not inconsistent with the object constraints of *Proceedings*. However, consider the global object  $\hat{O}'$  representing the remote object  $O' : Proceedings$ , where  $Sim(O',RefereedPubl)$  holds. Thus, in the integrated view  $\hat{O}'$  also belongs to *RefereedPubl*. However, since we know that  $O'$  satisfies both the intraobject condition and object constraint *oc2* of *Proceedings*, we can deduce the derived object constraint *rating*  $\geq 7$  on  $O'$ . Thus a potential discrepancy with object constraint *oc1* of *RefereedPubl* has been identified. Section 5 discusses how such discrepancies should be dealt with.  $\square$

### 4 Conformation of constraints

The phases of conformation and merging are distinguished in our discussion of constraints, too. In this section, we briefly discuss the former of these phases.

Local and remote constraints cannot be compared properly without assuring that they are expressed in compatible terms. Thus, conversions applied to objects and properties in the conformation phase must be propagated to the formulation of constraints. This is called semantic normalisation in [RPG95]. In conforming local and remote constraints, we distinguish the following subtasks:

#### 1. Allocating constraints to conformed classes

Due to structural conflicts between databases, the allocation of constraints to a conformed class takes some consideration. The conformed constraint may be allocated to a new virtual class which was created when converting values to objects to deal with object-value conflicts between databases. On the other hand, the conversion of objects to values (hiding of objects) leads to the hiding of constraints that involve properties that are not included in the (complex) values thus obtained.

#### 2. Attribute substitution

Conformed attribute names must be substituted for locally adhered attribute names. The type of a property may change in the conformation, too.

### 3. Domain conversion

Whenever a particular value for a property is used in a constraint, this value must be converted to a value of the domain of the corresponding conformed property.

### 4. Derived attributes

In the conformation phase, derived attributes may be introduced to deal with non-trivial property equivalence. Constraints may be derived for such attributes as well.

#### Example

In our example, consider the constraint *oc2* on *Publication*. In the remote database, a publisher is considered to be an object itself rather than just a value used in the description of a publication. This point of view is adhered to in the integration as well. As a result of this, the value of the local attribute *Publication.publisher* is converted to a virtual object in the conformation phase, and classified under a virtual local class *VirtPublisher*. In terms of the conformed classes, the constraint contains properties of *VirtPublisher*. Thus, the formulation of the constraint becomes:

**object constraint on *VirtPublisher*:**  
*oc1*: name in KNOWNPUBLISHERS

Moreover, consider the constraint *oc1* on *RefereedPubl*. The attribute 'rating' is involved in the property equivalence assertion  $propeq(ScientificPubl.rating, Proceedings.rating, multiply(2), id, avg)$ . The conversion function  $multiply(2)$  from the local scale to the remote scale must be applied to the rating value featured in the constraint. We thus obtain the following conformed constraint formulation:

**object constraint on *RefereedPubl*:**  
*oc1*: rating  $\geq 4$

$\square$

### 5 Constraints and the merging phase

Expressed in compatible terms, local and remote constraints may now be compared. This is the main subject of this paper, to be discussed in the present section. We first discuss the notions of subjectivity and objectivity of constraints; subsequently we discuss their implications on the two questions presented in the introduction.

## 5.1 Objectivity and subjectivity

The notions of objectivity and subjectivity are associated with the nature of a real-world phenomenon and the way in which a certain subject perceives this phenomenon. Their relevance to database interoperation is obvious once we realise that a database is nothing but a model of a real-world application domain, and that different databases will model a common application domain in a different way.

A database modelling assertion, such as a property value or a constraint, is called *objective* iff its validity is independent of the implicit assumptions made within the context of a particular database; otherwise it is called *subjective*. In the next subsections, we make these notions more precise for constraints and properties in an interoperation environment, respectively, and discuss the relationship between constraint subjectivity and value subjectivity.

### 5.1.1 Objectivity and subjectivity of constraints

In the context of database interoperation, we need to reconsider the nature of constraints defined on a database. In particular, within a *single* database environment, the following two statements are considered to be roughly equivalent.

- A constraint is a representation of an axiom that is valid in the part of the real world modelled by the database.
- A constraint is a restriction describing those database states that the designer considers correct ones.

In the context of database *interoperation* it is necessary to distinguish between these two constraint paradigms. This is due to the fact that we are dealing with a real-world situation modelled by different database designers, introducing the notion of subjectivity into the interpretation of database constraints.

#### Example

Referring to Figure 1, an example of an objective constraint would be *ocl* of class *Proceedings*. Although defined in database *Bookseller*, this constraint may be assumed to represent an assertion that is true beyond the context of this specific database. Of course, the modelling context influences the way the constraint is expressed, but not its validity. This is the kind of constraints dealt with in existing work [AQF95, RPG95]. On the other hand, consider constraint *cc2* of class *Publication*. This is a subjective constraint. It must be satisfied within the context of our library, but not necessarily by the integration. □

The question is then how to determine whether a constraint is objective or subjective. To answer this, we first discuss subjectivity of properties.

### 5.1.2 Objectivity and subjectivity of properties

Subjectivity and objectivity can also be associated with properties. To see what we mean by a subjective property, we return to the notion of property equivalence. Remember that equivalence of local and remote properties is denoted by assertions of the form *propeq*(*C.p, C'.p', cf, cf', df*), where *df* is a decision function determining a global value for the property given a local and a remote one. Decisive for the subjectivity of a property is the extent to which the decision function accounts for possible *value inconsistencies*. In particular, we distinguish four types of decision functions depending on their consequences for the subjectivity of the properties involved:

#### 1. Conflict ignoring function

This represents the situation where the decision function does not deal with possible value conflicts. That is, non-deterministically any of the values is chosen (denoted in the example by the *any* function). *Both* of the local and remote properties are regarded to be *objective*. Thus, *Publisher.name* and *Publication.publisher* are considered objective in our example specification.

#### 2. Conflict avoiding function

Here one of the equivalent properties is chosen as the most reliable source of values for the integrated property (the function *trust* in our specification). That is, one of the *properties* is considered *objective*. Thus, *Publication.ourprice* is seen as objective, whereas *Publication.shopprice* is subjective in our example specification.

#### 3. Conflict settling functions

The conflict is settled by picking one of the values using a certain decision procedure. Thus, one of the local and remote *values* is seen as *objective*, but in general we do not know beforehand from which of the equivalent properties it stems. Therefore we regard both *properties* as subjective. Examples of such functions are *max* and *min*.

#### 4. Conflict eliminating functions

The conflict is eliminated by defining a global value which is derived from both values. Note however that we demanded that  $df(a, a) = a$ . Examples are *avg* and *union*. Here *both* the local and the remote value are regarded as *subjective*. Thus, both *ScientificPubl.rating* and

Proceedings.rating are seen as subjective in our example specification.

### 5.1.3 Subjective constraints and subjective properties

A crucial question is now of course: what is the relationship between subjectivity of properties and subjectivity of constraints? To answer this question, consider the following example.

#### Example

Consider the *identical* conformed constraints ocl of classes **Publication** and **Item** (recall that 'ourprice' and 'libprice' are equivalent properties). Note that in the presence of the decision function *trust*, the following intuition concerning a constraint  $\phi$  is **not** valid.

$(DB \text{ satisfies } \phi \wedge DB' \text{ satisfies } \phi) \Rightarrow DB \text{ satisfies } \phi.$

This can easily be seen by assuming the (libprice, shopprice) values of a particular book provided by **Publication** and **Item** are (26,29) and (22, 25), respectively, and applying the decision functions *trust*(CSLibrary) resp. *trust*(Bookseller) for 'libprice' and 'shopprice'. Apparently, the value subjectivity incurred by the decision functions yields this object constraint to be subjective, even if it is defined in *both* component databases.  $\square$

In general, it is a design decision which local and remote constraints are considered to be valid within the database's partial view of the world only (subjective constraint), and which are also adhered to in the context of the integrated view (objective constraint). However, as can be concluded from the example, an integration specification is consistent only if

*Subjectivity of values implies subjectivity of constraints.*

That is, constraints involving subjective properties are necessarily subjective themselves. Note that the implication is one-directional; constraints may be declared subjective even though they involve objective properties only.

## 5.2 Constraints and the integrated view

Recall that in the introduction we distinguished two main questions, regarding the formulation of global integrity constraints given the local and remote ones, and the detection of possible inconsistencies between integrity constraints and the specification of the integrated view. We here refine this to the following three issues.

- **How is the set of integrated constraints determined?**

This question is related to the notion of subjective and objective constraints. For objective constraints, the natural answer is that the set of global constraints is the union of locally and remotely defined constraints. For subjective constraints, global constraints must be *derived* from local and remote constraints depending on how the state of the global view is derived from the local and remote database state.

- **When is there a conflict?**

As we shall see, the answer to this question is dependent on the type of constraint and the type of object relationship involved.

- **What should be done if a conflict is found?**

The choice here is either to disqualify some of the existing constraints or to adapt the specification of the integration. In other words, the specification of the integration may be considered invalid if it leads to conflicting constraints on the integrated view.

### 5.2.1 Integration of object constraints

The integration of object constraints stemming from different databases is strongly influenced by the relationship between the objects on which the constraints have been defined. In the following, we assume that  $O$  is a local object on which the set of object constraints  $\Omega$  has been defined, and  $O'$  is a remote object with object constraint set  $\Omega'$ <sup>2</sup>. Note that a possible descriptivity relationship between  $O$  and  $O'$  is dealt with in the conformation phase. Hence we here discuss the cases of equality, strict and approximate similarity. Note furthermore that object constraints discussed here include the derived constraints discussed in Section 3.

#### Object equality

At first glance, one would expect equality of objects to suggest equivalence of their object constraints. It follows from our discussion of subjectivity, however, that this is not necessarily the case. We assume that the sets  $\Omega_s, \Omega_o, \Omega'_s$  and  $\Omega'_o$  have been specified, representing the set of subjective local object constraints, objective local object constraints, subjective remote object constraints and objective remote object constraint, respectively. We now discuss their integration with reference to the three major issues identified above.

The **integrated object constraints** are determined as follows. It follows from our discussion that

<sup>2</sup>A set of object constraints defined on an object represents the conjunction of these constraints.

all objective object constraints are integrated object constraints. Moreover, from subjective local and remote object constraints, integrated object constraints may be derived. The general problem of deriving a global object constraint  $\phi$  given local and remote constraints  $\phi, \phi'$  and a decision function  $df$  is beyond the scope of this paper. We here restrict ourselves to the definition of some necessary conditions under which such a derivation is possible.

Let  $\phi$  be a *normalised* subjective object constraint, meaning that  $\phi$  cannot be written as  $\phi_1 \wedge \phi_2 \dots \wedge \phi_n$  (constraints of such a form are normalised into  $n$  separate object constraints). A normalised object constraint defines a correlation between the values of the properties involved in the constraint. Let  $\Xi(\phi)$  be the set of subjective properties constrained by  $\phi$ . The following conditions must be true for  $\phi$  to be involved in the derivation of a global constraint:

(1) None of the decision functions defined on  $\Xi(\phi)$  must be conflict avoiding. If a property  $p \in \Xi(\phi)$  is subjective due to a conflict avoiding function, the value of  $p$  plays no role in the determination of the value of the global property  $\dot{p}$ . Hence restrictions on  $p$  cannot be propagated to restrictions on  $\dot{p}$ . Since the values of  $\Xi(\phi)$  are correlated by  $\phi$ , none of the restrictions on properties in  $\Xi(\phi)$  can be propagated.

(2) If a property  $p \in \Xi(\phi)$  has a conflict settling function defined on it, then the derivation of a global constraint  $\dot{\phi}$  from  $\phi$  must involve a remote constraint  $\phi'$  such that  $p' \in \Xi(\phi')$ . The determination of the value of  $\dot{p}$  involves a comparison of the values of  $p$  and  $p'$ . Thus, restrictions on  $p$  can only be propagated to restrictions on  $\dot{p}$  if comparable restrictions exist for  $p'$ .

#### Example

Consider a local object  $O$ :`ScientificPubl` and a remote object  $O'$ :`Proceedings` such that  $Eq(O, O')$  holds. Now consider  $\phi$  : `rating >= 4`, a conformed subjective object constraint on  $O$ , and  $\phi'$  : `publisher.name = 'ACM' => rating >= 6`, a subjective object constraint on  $O'$ . The decision functions for `Publisher.name` and `Proceedings.rating` are *any* and *avg*, respectively, yielding the former property to be objective, and the latter subjective. The global object constraint

`publisher.name='ACM' implies rating >= 5` can be derived. On the other hand, consider the object constraints `ocl` of classes `Publication` and `Item` discussed in a previous example. The conflict avoiding decision functions on `'shopprice'` and `'libprice'` render both of these constraints subjective, and no global object constraints can be derived from them.  $\square$

A **conflict** between local and remote object constraints is found if the set of integrated object con-

straints is inconsistent, i.e.  $\dot{\Omega} \models false$ . This is what we might call an *explicit conflict*. An *implicit conflict* occurs if the state of a global object  $\dot{O}$  does not satisfy its integrated object constraints. It can easily be seen that this situation can only occur for an objective object constraint  $\phi$  involving at least one property  $p$  which is equivalent to a property  $p'$ , for which a conflict ignoring function has been defined. The global value of  $\dot{O}.p$  may then be obtained from the remote property  $p'$  due to the non-determinism introduced by such functions. Now if an equivalent constraint  $\phi'$  was not defined on  $p'$ ,  $\dot{O}$  will not necessarily satisfy  $\phi$ , even though  $\phi$  is regarded to be objective.

This raises the issue of **resolving a conflict** once it has been detected. We identify three options.

#### 1. Change or ignore local and/or remote constraints

This straightforward solution is the only one mentioned in current literature. Note that in our context, changing local constraints would mean changing their specified status from objective to subjective.

#### 2. Change the object comparison rules

This is based on the idea that conflicting constraints indicate that objects  $O$  and  $O'$  are not truly equivalent, and thus the object comparison rules are incorrect, and should be adapted.

#### 3. Change the decision functions

Changing a decision function affects the set of global integrity constraints involving that property that can be derived. Thus an inconsistency in the global object constraints can be resolved.

#### Strict similarity

The case of (strictly) similar objects should be distinguished sharply from equality of objects. In particular, property subjectivity does *not* play a role with similar objects, since decision functions apply only to properties of equivalent objects. Therefore, the designer has more freedom to label object constraints as subjective or objective.

Regarding the issues identified before, the **integrated object constraints** are simply the union of the objective local and remote object constraints. Subjective constraints do not contribute to the integrated view.

With strictly similar objects, we have to be very strict in defining constraint **conflicts**. In fact, we define a constraint conflict to arise whenever  $\Omega' \not\models \Omega$ . That is, in order for remote objects  $O'$  to be added to a class  $\dot{C}$ , they must satisfy all object constraints associated with  $\dot{C}$ .

Resolving such conflicts is done by adding object constraints of  $C$  as *intraobject conditions* on  $O'$  to the condition part of the rule describing the relationship  $Sim(O', C)$ . An additional rule defining approximate similarity for objects not satisfying these additional conditions may be added.

### Example

Consider the object comparison rule  $Sim(O':Proceedings, RefereedPubl) \leftarrow O'.ref? = true$ . In Section 3, we discussed how from the intraobject condition  $O'.ref? = true$  and the object constraint  $oc2$  of *Proceedings* the derived constraint  $\phi : rating \geq 7$  on  $O'$  could be deduced. Now since  $\phi \Rightarrow rating \geq 4$ , which is the conformed form of object constraint  $oc1$  of *RefereedPubl*, it is assured that  $O'$  is a valid *RefereedPubl* (note that the inherited object constraint  $oc1$  of *Publication* is satisfied as well).

On the other hand, suppose  $oc2$  of *Proceedings* were  $ref? = true \Rightarrow rating \geq 3$ . In that case, the derived constraint would be  $\phi : rating \geq 3$ . Since then  $\phi \not\Rightarrow rating \geq 4$ , the object comparison rule would have to be changed into  $Sim(O':Proceedings, RefereedPubl) \leftarrow O'.ref? = true \wedge O'.rating \geq 4$ .  $\square$

### Approximate similarity

In cases of approximate similarity, we cannot really speak of object constraint **conflicts**, as objects  $O : C$  and  $O' : C'$  with possibly conflicting constraints will end up in different global classes anyway. The only additional **integrated object constraints** that can be determined are those to be defined on their common virtual superclass  $CV$ . This is simply the disjunction of  $\Omega$  and  $\Omega'$ .

In fact, we may have that  $\Omega \models \neg\phi'$ , where  $\phi' \in \Omega'$ . This corresponds to the case where the classes  $C$  and  $C'$  represent different *horizontal fragments* of their common superclass  $CV$ , where the membership condition is  $\phi'$ .

#### 5.2.2 Integration of class constraints

As classifications themselves are inherently subjective, so are class constraints. Class constraints are typically valid for the local extension of a particular class. We therefore adhere to the default assumption that class constraints cannot be propagated to the integrated view.

An exception is any class for which neither object equivalence rules nor strict similarity rules have been defined. Such a class has what we might call *objective extension*. As the global extension of such a class is equal to its local extension, it is assured that all class constraints remain valid for the integrated view.

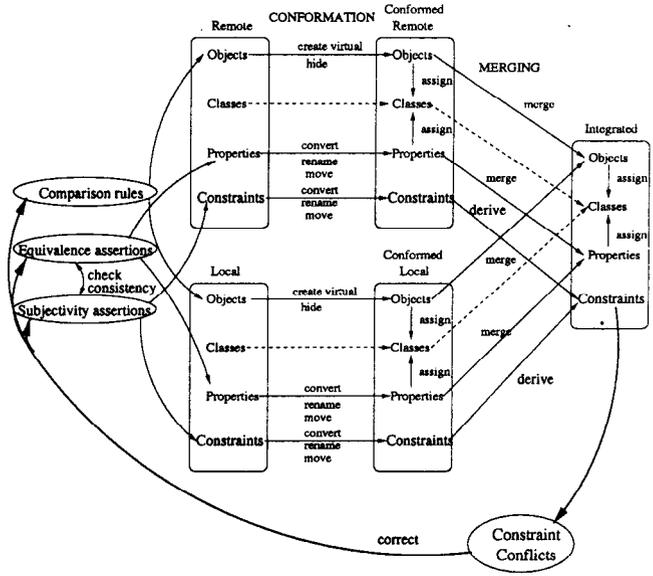


Figure 3: Inclusion of constraints into our methodology

If a local class  $C$  does not have objective extension, but a class constraint  $\phi$  of  $C$  is still considered objective, it must either be provable that the addition of a remote object  $O'$  to  $C$  will never lead to a violation of  $\phi$ , or any addition of such an object that leads to a violation of  $\phi$  must be rejected by a global integrity enforcing mechanism. In this paper, we are primarily concerned with the definition of global constraints that are an immediate consequence of the locally enforced constraints, however. Hence we assume by default that class constraints are subjective.

This sharp distinction between object constraints and class constraints comes as no surprise if one realises that unlike object constraints, class constraints are not inheritable. A notable exception is the **key constraint**. There is an analogy of this exception for the case of database interoperation. If all equality rules defined on a class  $C$  are of the form  $Eq(O : C, O' : C') \leftarrow O.k = O'.k'$ , where  $k$  is a key of  $C$  and  $k'$  is a key of  $C'$ , and similarity rules are defined only for objects of classes for which equality rules exist as well, then the key constraint on  $C$  is still valid for the integrated view.

#### 5.2.3 Integration of database constraints

Database constraints should be regarded as subjective constraints. The complications of regarding a local database constraint as objective are immense, as illustrated by our example database constraint of Figure 1. A discussion hereof is beyond the scope of this paper.

## 6 Conclusion

In this paper, we have identified two roles of integrity constraints defined on interoperable component databases.

The first role is as a basis for specification of constraints on the integrated view, thus supporting global query and transaction processing. We discussed the notion of objectivity and subjectivity of component constraints and how these notions influence the formulation of global constraints.

The second role is as a semantic check on the validity of the specification of an integrated view. We illustrated how inconsistencies between local constraints the specification of the integration can be detected, and how this specification can subsequently corrected.

Figure 3 illustrates the role of constraints in our methodology. From our discussion, we conclude that the incorporation of constraints into existing integration methodologies is feasible, if not essential. The ideas developed in this paper form the basis for the development of design tools that assist in determining the effects of design decisions made during the construction of integration specifications on the constraints that will be valid on the integrated view resulting from such a specification. Constraint conflicts detected can be used to highlight errors in the specification, and suggestions can be done to the user as to how to correct them.

## References

- [AQF95] R. M. Alzahrani, M. A. Qutaishat, N. J. Fiddian & W. A. Gray, "Integrity merging in an object-oriented federated database environment," in *13th British National Conference on Databases, Manchester, UK*, Springer-Verlag, New York-Heidelberg-Berlin, 1995, 226-248, LNCS #940.
- [BBZ93] H. Balsters, R. A. de By & R. Zicari, "Typed sets as a basis for object-oriented database schemas," in *Proceedings Seventh European Conference on Object-Oriented Programming, July 26-30, 1993, Kaiserslautern, Germany*, LNCS #707, O. M. Nierstrasz, ed., Springer-Verlag, New York-Heidelberg-Berlin, 1993, 161-184, See also <http://wwwis.cs.utwente.nl:8080/oodm.html>.
- [BLN86] C. Batini, M. Lenzerini & S. B. Navathe, "A comparative analysis of methodologies for database schema integration," *ACM Computing Surveys* 18 (December 1986).
- [CGW96] S. S. Chawathe, H. Garcia-Molina & J. Widom, "A toolkit for constraint management in heterogeneous information systems," in *12th International Conference on Data Engineering, New Orleans*, IEEE Press, Montvale, NJ, 1996.
- [DaH84] U. Dayal & H-Y. Hwang, "View definition and generalization for database integration in a multidatabase system," *IEEE Transactions on Software Engineering* 10 (November 1984), 628-645.
- [DKM93] P. Drew, R. King, D. McLeod, M. Rusinkiewicz & A. Silberschatz, "Report of the workshop on semantic heterogeneity and interoperation in multidatabase systems," *SIGMOD RECORD* 22 (September 1993), 47-55.
- [FKS94] J. Flokstra, M. van Keulen & J. Skowronek, "The IMPRESS DDT: A database design toolbox based on a formal specification language," in *Proceedings ACM-SIGMOD 1994 International Conference on Management of Data*, ACM Press, New York, NY, 1994, 506.
- [GPQ95] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman & J. Widom, "The TSIMMIS approach to mediation: Data models and languages," Stanford University, Stanford, CA, 1995.
- [GrW96] P. Grefen & J. Widom, "Protocols for integrity constraint checking in federated databases," in *COOPIS'96, Brussels, Belgium*, IEEE Press, Montvale, NJ, 1996.
- [LSP93] E-P. Lim, J. Srivastava, S. Prabhakar & J. Richardson, "Entity identification in database integration," in *Proceedings Ninth International Conference on Data Engineering, Vienna, Austria, April 19-23, 1993*, IEEE Computer Society Press, Washington, DC, 1993, 294-301.
- [RPG95] M. P. Reddy, B. E. Prasad & A. Gupta, "Formulating global integrity constraints during derivation of global schema," *Data & Knowledge Engineering* 16 (1995), 241-268.
- [ShL90] A. P. Sheth & J. A. Larson, "Federated database systems for managing distributed, heterogeneous and autonomous databases," *ACM Computing Surveys* 22 (September 1990), 183-236.
- [VeA96] M. W. W. Vermeer & P. M. G. Apers, "On the applicability of schema integration techniques to database interoperation," in *Proceedings Fifteenth International Conference on Conceptual Modelling (ER'96), Cottbus, Germany*, Springer-Verlag, Berlin, 1996.
- [VeA95] M. W. W. Vermeer & P. M. G. Apers, "Reverse engineering of relational database applications," in *Proceedings Fourteenth International Conference on Object-Oriented and Entity-Relationship Modeling (ER'95), Gold Coast, Australia*, M. P. Papazoglou, ed., Springer-Verlag, New York-Heidelberg-Berlin, December 1995, 89-100, LNCS #1021.