# CONFLICTS AND CORRESPONDENCE ASSERTIONS IN INTEROPERABLE DATABASES

**Stefano SPACCAPIETRA**
Ecole Polytechnique Fédérale - DI - LBD
IN/Ecublens - CH 1015 Lausanne
*spaccapietra@elma.epfl.ch*

**Christine PARENT**
Université de Bourgogne - Dép. Informatique
B.P. 138 - F 21004 Dijon Cedex
*badine@frccub11.bitnet*

## 1. Introduction

Irrespective of the approach (integrated, federated, or multibase [5, 8]) taken for the interoperability in database systems, the designers are faced with the problem of comparing the information content of the various interconnected databases. Two questions arise:
- how does one know if and to what extent the databases share related information?
- if so, how can one instruct the system about commonalities, so that the system can manage a global schema of the underlying databases?

It is not possible to produce a definite answer to the first question automatically by searching the various schemas and the corresponding databases. Even if a complete match is found (i.e. both at schema and instances levels), there is never an absolute guarantee that the match will always be verified in the future. Nevertheless, automatic tools may efficiently assist the database administrator (DBA) in the task of identifying commonalities among schemas. Several such tools have been developed [2, 3, 10]. They act as "investigators", trying to find out which elements from the two schemas show enough similarity (in terms of names, domains, components, relationships to other elements, ...) to be candidate for a match. The DBA is then prompted to confirm or deny the match.

An alternative method is to assume that the DBA knows about the commonalities. In this case, the DBA directly forwards this information to the system, usually as correspondence assertions: each particular assertion states some relationship between two elements and their instances in two databases.

The knowledge expressed by these assertions or by confirmed matches is used by the system either to generate mappings between corresponding elements (and the queries upon), or to appropriately merge these elements into an integrated schema, (and build the mappings between the resulting schema and the input schemas). In the latter case the system acts as an "integrator", achieving a single description of the knowledge from the several input schemas[1].

Correspondence assertions, whether stated by the DBA or found with the help of an investigator, should be able to resolve all discrepancies among schemas and among databases. This is not the case with current methodologies and tools. The next section proposes a taxonomy that identifies where current technology fails to automatically solve conflicts. Extensions needed to overcome these limitations are introduced in sections 3 and 4.

## 2. A taxonomy for inter-schema discrepancies

There has been a large amount of work in database interoperability, with some emphasis on schema integration issues. Good surveys may be found in [1, 8]. However, the terminology is still somewhat confusing. The following taxonomy stems from the various choices that a designer has to make in the process of schema definition.
Let us assume that two designers are looking at overlapping universes of discourse (UoDs). Differences may appear at any of the following design steps which give rise to a variety of conflicts.

---

[1] Alternatively, the DBA may define a program to map each element into the other, or to build the integrated schema. The role of the system is then limited to supporting a set of restructuring operations the DBA can use [6].

*1. Select and classify real world objects from the UoD.*

The two designers may not perceive exactly the same sets of objects, or adopt different classifications. This will result in overlapping sets of objects. For instance, a designer can group all students into a "Student" object class, while the other designer can identify a "CS-Student" class, grouping students majoring in computer science, as the one (s)he is interested in.
The situation where two elements from two schemas represent sets of UoD objects which are related by a set comparison operator other than equality is called a **semantic conflict.**

This kind of conflicts has been extensively dealt with in the literature. The generalization concept is used to resolve such conflicts. For instance, the "CS-Student" class will be defined as a subclass of the "Student" class.

*2. Attach properties to the selected classes.*

The two designers may not perceive exactly the same set of properties for related classes. For instance, let us assume two relational schemas, S1 and S2, both describing the same set of expensive car models:

S1: Expensive_car (modelname, manufacturer, maximumspeed, price)
S2: Car_model (name, horsepower, fuelconsumption, price)

S1 and S2 designers selected different items for record keeping, because of their different interest in the many forms of the available information on car models in the real world (one designer may have to keep data for advertisements in a fine arts journal, while the other is concerned with advertisements in a technical journal, for instance).
The situation where two elements, representing related sets of UoD objects, are described with different sets of properties is called a **descriptive conflict.**

Descriptive conflicts include naming conflicts, due to homonyms and synonyms, as well as conflicts on attribute domain, scale, cardinalities, constraints, operations, et cetera [4, 9].

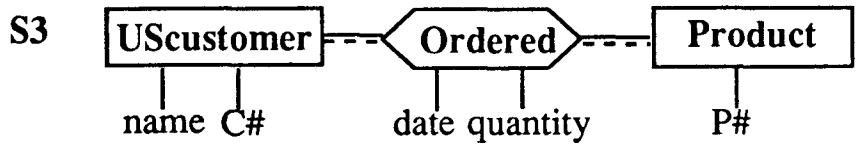
*3. Choose a data model as representation vehicle.*

The two designers may use different data models. For example, one may choose a relational model and other an object oriented model. The situation where two schemas are defined with different data models is called a **data model conflict.**

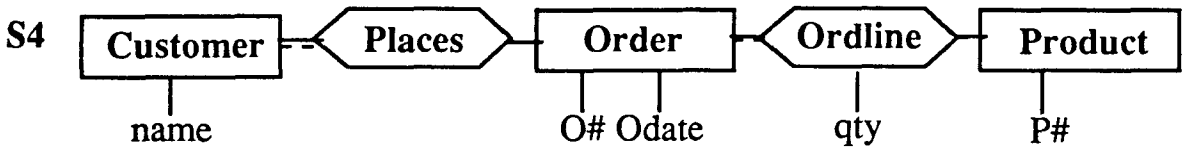
*4. Define data structures representing the UoD.*

Even if the two designers use the same data model, they can choose different constructs to represent common UoDs. In object-oriented models, for instance, a designer willing to describe a component X of an object type O, may choose between creating a new object type X (and include a reference to X in O) or adding an attribute X to O. The situation where two related UoDs are represented using different data structures is called a **structural conflict.**

The conflicts defined above are orthogonal and can be cumulated. Indeed, discrepancies between schemas usually show a mix of conflict types. Different data models, or different sets of UoD objects generate different structures. As an example, consider the following entity-relationship (ER) diagrams[2] which represent related UoDs:




[2] diagrams are drawn according to ERC+ rules. A single plain line denotes a 1:1 cardinality. Combination of a plain and a dotted line denotes a 1:N cardinality. ERC+ is an extended ER model supporting complex objects and object identity [7].

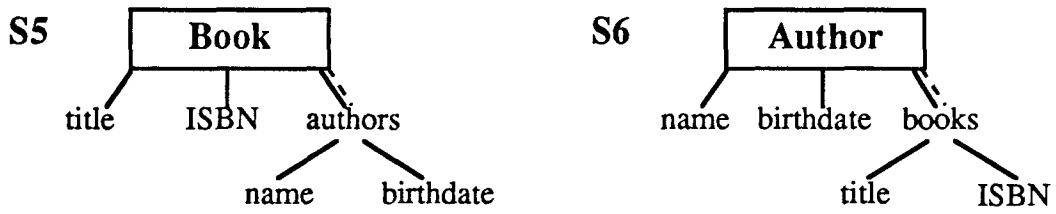



Both represent some information about customers who can order products from an enterprise. S4 includes all customers, while S3 only considers a subset of the customers (semantic conflict). Customer's attributes differ between S3, S4 (descriptive conflict). The ordering information is given in S3 as a direct relationship between a customer and a product (s)he ordered. S4 favoured a more detailed representation based on order's materialization as an Order entity type (structural conflict).

While semantic and descriptive conflicts have been largely investigated in the field of assertion based methodologies, data model and structural conflicts remain to be satisfactorily addressed.
Heterogeneity of data models has been tackled by assuming that in a preliminary step all input schemas are translated into some common data model.
As for structural conflicts, an almost unanimous assumption is that they have to be solved manually by the DBA. (S)he is in charge of replacing conflicting structures with equivalent non conflicting ones (a schema modification process known as schema conforming). Integration then proceeds with the new input schemas. Usually, the original input schemas are lost.
We believe that better solutions for both conflict types may be achieved by developing a more powerful assertional approach. We discuss this in the coming sections.

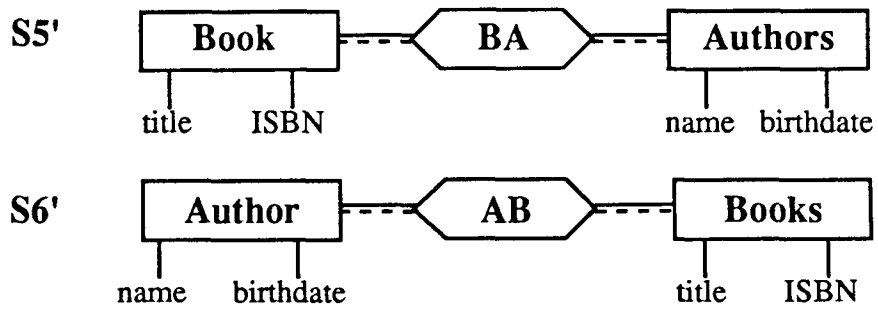
## 3. Correspondence assertions and structural conflicts

The extent to which structural conflicts may arise depends on the data model in use. The more constructs it supports, and the more complex they are, the more potential for conflicts it bears.
In the ER approach, for instance, the designer is left with the responsibility to decide whether a real world object should be represented as an entity, a relationship, or an attribute. In object-oriented models, there are at least six possible ways of representing an association between two object classes: embedding one class as an attribute to the other, using single or cross referencing, and creating an additional object to represent the association.
The following diagrams (S5 and S6) illustrate two different ERC+ [7] representations compatible with the same universe of discourse (a library).
The viewpoint represented in S5 considers authors as a (multivalued) attribute of the Book entity type. Conversely, the viewpoint represented in S6 considers books as a (multivalued) attribute of the Author entity type. The two schemas show a structural conflict.




Correspondence assertions, as currently proposed, are of no help in this particular case. They are only able to relate comparable constructs: an entity type from one schema compared to an entity type from another schema, a relationship type compared to a relationship type, an attribute compared to an attribute. No mixed comparison (an attribute with an entity type, for instance) is supported. The consequence is poor interoperability and error-prone techniques, as shown in [11].

With current methodologies, there is nothing which can be asserted between S5 and S6. An integration process instructed to merge S5 with S6 would produce a schema showing both entity types, just as they are. To avoid such an incorrect result, the DBA has to modify the input schemas before integration is activated. The DBA may choose to replace S5 with S6 (or S6 with S5). Alternatively the DBA may change both schemas, replacing S5 with S5' and S6 with S6'. In both cases the initial specifications are abandoned to make the integration possible.




The reason to modifiy the input schemas in case of structural conflicts, is the inadequacy of mapping capabilities, between corresponding elements or among input schemas and the integrated schema. If more sophisticated mapping capabilities are available (like the ability to transform an object into an attribute, and viceversa), the burden of solving structural conflicts can be taken over by the system. Consequently we propose to extend the scope of correspondence assertions to support all four kinds of conflicts. For instance, if S5 and S6 refer to the same UoD, the following two assertions describe the correspondences between their elements:

Book ≡ Author•books  with corresponding attributes: title=title, ISBN=ISBN

Book•authors ≡ Author  with corresponding attributes: name=name, birthdate=birthdate

These assertions state that: i/ there is a structural conflict in the representation of books and authors; ii/ there is no semantic conflict concerning books and authors (otherwise, instead of ≡, the DBA would state either one of $\subseteq$, $\subset$, ..., $\cap$, $\neq$); iii/ there is no descriptive conflict, as books (respectively authors) have the same attributes in both schemas, once the conflicting attribute authors (respectively books) is transformed into an entity type.

Generally speaking, structural conflicts imply that a correspondence may exist between two schema elements whether they are an object class, an attribute, a relationship, et cetera. Assertions may thus be classified as:

- correspondence assertions between elements of the same type (i.e. elements defined with the same modelling concept).

  Example (S5'-S6'): Book ≡ Books  Authors ≡ Author

  This is the type of assertions currently supported in literature.

- correspondence assertions between elements of different types.

  Example (S5-S6): Book ≡ Author•books  Book•authors ≡ Author

  These assertions state that the real world represented by the entity type Book in S5 is the same as the one represented by the books attribute in S6 (same for authors).

  In schema integration, for instance, the processing of these assertions leads to an automatic schema conforming step (equivalent to what a DBA would do manually). This builds the intermediate internal representations which are finally merged to produce the integrated schema. Mappings are generated among the integrated schema and the original input schemas. Keeping the input schemas unchanged is of particular relevance to database integration.

- correspondence assertions between links.

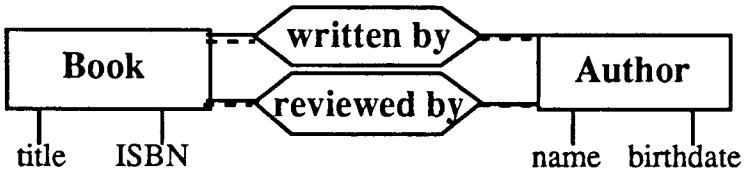
  The term link is used here to refer to a connection between two elements in a schema, irrespective of the data model. Depending on the data model, it may be a connection between an attribute and its parent element, a connection between two object types through a reference, or a connection between a relationship type and an entity type through a role.

  Element correspondence assertions are not sufficient to completely define commonalities between two schemas. For instance, the following correspondence assertions for S5-S6:

  Book ≡ Author•books  Book•authors ≡ Author

do not imply that the association between books and authors is the same. One could perfectly assume that S5 talks about authors having written the associated book, while S6 describes for each author the books (s)he has reviewed. The integrated schema in this case should be S7:

S7




This is the schema which could be generated by default based on the two previous assertions describing the correspondences between the elements of S5 and S6 (except that relationship names would be the standard default names).
Assume now that for both S5 and S6, the semantics of the book-author link is the same. S5 gives for each book its authors, S6 gives for each author the books (s)he has written. This identity has to be explicitly stated by an assertion about links:

Book — authors ≡ books — Author

This additional knowledge leads the integrator to produce S8 instead of S7:

S8


The semantics of correspondence assertions can be defined by refering to the real world counterpart of the involved elements. Larson et al [4] defined the real world state of an object class O, RWS(O), as "the set of real world instances of object class O at a given moment in time". The RWS concept needs to be extended to cover attributes and links.
The definition of the RWS of an object type may be easily extended to an attribute. We define the RWS of an attribute as the set of real world objects that the set of present values of the attribute represent (as suggested in [9] this definition abstracts from attribute descriptors). Having similar definitions allows one to state a correspondence assertion between an object type and an attribute.
On the other hand, a link X — Y is a connection between elements X and Y. Its RWS is made up of pairs of real world objects, one from RWS(X) and one from RWS(Y), such that the two objects in the real world are bound by an association represented in the database by the X — Y link.

## 4. Correspondence assertions and data model conflicts

There is no reason why a correspondence assertion should not relate two elements from heterogeneous schemas (assuming the mapping between the underlying data models is known). In our approach to schema integration, the semantics of such assertions is properly understood by refering to a generic data model, specifically designed to support the reasoning about the integration [12]. Generic integration rules have been defined by using the concepts of the generic data model, and then tailored to different data models which might support the input schemas.

For instance, assume the following object-oriented structure of an Author class:

**S9:** **Class** Author **tuple** < name : • • •, birthdate : • • •,
books: **setof tuple** < title: • • •, ISBN: • • • >>

A comparison between the ER schema S5 and the object-oriented schema S9 results in the same assertions as before:

Book ≡ Author•books with corresponding attributes: title=title, ISBN=ISBN

Book•authors ≡ Author with corresponding attributes: name=name, birthdate=birthdate

Book — authors ≡ books — Author

If an object-oriented result is requested, the integrator generates the following integrated schema:

**S10:** **Class** Author **tuple** < name : • • •, birthdate : • • •, books: **setof** Book>
**Class** Book **tuple** < title: • • •, ISBN: • • •, authors: **setof** Author>

## 5. Conclusion

This short position paper:
- proposed a taxonomy of conflicts arising from schema comparisons,
- emphasized the inadequacy of current methodologies to cope with data model and structural conflicts,
- outlined the extensions to correspondence assertions which are needed to cover the above types of conflicts,
- introduced the extensions to the real world state concept needed for proper understanding of the correspondence assertions.

### Acknowledgements

The authors are indebted to Prof. Bharat Bhargava, as well as to the guest editor and the reviewers, for helpful suggestions which significantly aided in improving this paper.

## References

[1] C. Batini, M. Lenzerini, S.B. Navathe: "A Comparative Analysis of Methodologies for Database Schema Integration", *ACM Computing Surveys*, Vol.15, n°4, December 1986, 323-364

[2] M. Bouzeghoub, I. Comyn-Wattiau: View Integration by Semantic Unification and Transformation of Data Structures, *Proceedings 9th International Conference on Entity-Relationship Approach*, Lausanne, October 8-10, 1990, 413-430

[3] S. Hayne, S. Ram: Multi-User View Integration System (MUVIS): An Expert System for View Integration, *IEEE 6th International Conference on Data Engineering*, Los Angeles, February 1990, 402-409

[4] J.A. Larson, S.B. Navathe, R. Elmasri: "A Theory of Attribute Equivalence in Databases with Application to Schema Integration", *IEEE Transactions On Software Engineering*, Vol. SE-15, n°4, April 1989

[5] W. Litwin, L. Mark, N. Roussopoulos: Interoperability of Multiple Autonomous Databases, *ACM Computing Surveys*, September 1990, 267-293

[6] A. Motro: Superviews: Virtual Integration of Multiple Databases, *IEEE Transactions On Software Engineering*, Vol. SE-13, n° 7, July 1987, 785-798

[7] C. Parent, S. Spaccapietra: "ERC+: an object based entity-relationship approach", in *Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development*, P.Loucopoulos, R.Zicari Eds., John Wiley, 1992

[8] A.P. Sheth, J.A. Larson: "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Surveys*, Vol.22, n°3, September 1990, 183-236

[9] A.P. Sheth, S.K. Gala: "Attribute Relationships: An Impediment in Automating Schema Integration", *Proc. Workshop on Heterogeneous Database Systems,* Chicago, December 11-13, 1989

[10] A.P. Sheth, J.A. Larson, A. Cornelio, S.B. Navathe: A Tool for Integrating Conceptual Schemas and User Views, *IEEE 4th International Conference on Data Engineering*, Los Angeles, February 1-5, 1988, 176-183

[11] S. Spaccapietra, C. Parent, Y. Dupont: "View integration: a step forward in solving structural conflicts", to appear in *IEEE Transactions on Knowledge and Data Engineering* , October 1992

[12] S. Spaccapietra, C. Parent, Y. Dupont: "Model Independent Assertions for Heterogeneous Schema Integration", *LBD research report*, EPFL, Lausanne, February 1991, submitted to VLDB Journal