On spatial database integration

Thomas Devogele,
Institut Géographique National - COGIT, F 94100 Saint Mandé
thomas.devogele@ign.fr

Christine Parent, and Stefano Spaccapietra
Swiss Federal Institute of Technology, EPFL-DI-LBD, CH 1015 Lausanne
(parent, spaccapietra)@di.epfl.ch,
http://lbdwww.epfl.ch

Abstract. This paper investigates the problems that arise when application requirements command that autonomous spatial databases be integrated into a federated one. The paper focuses on the most critical issues raised by the integration of databases of different scales. A short presentation of approaches to interoperability and of the main steps composing the integration process is given first. Next, a general format is proposed for precisely defining correspondences between objects of two databases. The format can deal with a wide range of discrepancies in GIS data. Last, a solution is presented for aggregation conflicts which arise when one object of one database corresponds to a set of objects in the other database, a very frequent case when the databases are of different scales. The method is applied to excerpts of real cartographic databases.

1. Introduction

One of the major problems that the development of GIS applications is facing today is data acquisition. Not that data is not available: geographic data collection has been going on for centuries. Some of that is still stored on paper, including maps, some has been digitized and is stored in current GIS systems (at best) or in traditional files or databases. However, too often their reuse for new applications is a nightmare, due to poor documentation, obscure semantics of data, diversity of data sets (what information is stored, how it is represented and structured, what quality it has, which date it refers to, which scale is used, ...), heterogeneity of existing systems in terms of data modeling concepts, data encoding techniques, storage structures, access functionalities, etc.

Despite such obstacles, reusability of existing data is a must, simply because of the high cost of acquiring new geographical data from scratch. Once application requirements analysis has identified what data is needed, the modern designer has to look around to all data stores of potential interest to find out which ones may already have some of the data (s)he needs. Most likely, part of that will indeed exist, spread into pieces over various heterogeneous data stores, part of it will not exist and will have to be acquired. Eventually, the newly acquired data and the many pieces of reused data will be integrated into a single, uniform, non-redundant data store, which will serve as the underlying database for the new applications. The process of unifying existing data sources into a single framework is called database integration. It takes as input a set of databases (schemas and data instances), and produces as output a single unified description of the

input schemas (called the integrated schema) and the associated mapping information supporting integrated access to existing data instances through the integrated schema (Batini et al. 1986) (Parent et al. 1998). Please note that we use the usual database terminology. The term *data model* refers to the set of abstract data modeling concepts in use (e.g. object type, relationship type, attribute), otherwise termed the meta-model in the GIS community. The term *schema* refers to a description of application specific object types, relationship types, etc., for a given database (otherwise termed the data model). The term *data instance* refers to the data in the database which physically describes an application object or relationship.

Database integration is the most sophisticated and most powerful approach to data interoperability. Simpler alternatives exist and it is worthwhile mentioning them to provide a clear understanding of the issue. A very first basic approach, not attempting any integration, is to provide users with a global catalog of accessible information sources, where each source is described by some associated meta-data, e.g.: representation mode, scale, last update date, data quality level (Stephan et al. 1993) (Uitermark 1996). The Alexandria Digital Library project (Frew et al. 1995) is one of the major efforts to build sophisticated tools for such catalogs. A variant to this solution is the use of dedicated Web browser services to explore GIS data available at different sites (GEO2DIS 1997).

A first step into integration is to integrate the existing data by hand, specifying and implementing ad-hoc solutions. This may be done by splitting the new applications into pieces, so that each piece is tailored to access only one data store and to pass the local data in an appropriate format over to a global application which performs any global processing and synthesizes the result. Alternatively, the data of interest can be extracted from the local sources and copied through ad-hoc routines into a new single database, which is then made available to the new applications. An example of this approach is the European project MEGRIN (Illert and Wilski 1995). Both ways have evident drawbacks related to lack of scalability and consistency, and duplication.

The second path to interoperability is through standardization. The definition of standard data modeling and manipulation features provides a reference point which facilitates data exchange among heterogeneous systems. Two kinds of standards can be developed:

- data model standards specify which abstract modeling concepts have to be used. For non spatial data, standards exist for relational databases and are currently being developed for object-oriented databases (by ISO committees on SQL-3 and by the Object Management Group). Data model standards for spatial databases are being developed by ISO/TC 211 (ISO/TC 211 1996), CEN/TC 287 (CEN/TC 287 1996) and by the OpenGIS consortium (OGIS)).
- schema standards: these recommend a predefined data/process design (a template) for a specific application area, e.g., water management or facilities management. Such a standard provides a fixed schema in a given data model.

Standards, however, only define a target for data conversion. They do not address the problem of how to convert existing data into the standard format and how to integrate data from different sources.

The third alternative is to develop a software system to support data interoperability. Various solutions exist or are being investigated. The marketplace offers packaged gateways to connect different database management systems (DBMSs), mainly relational ones. Gateways allow one given system to access data from another given system. Some recent, more sophisticated products provide facilities for the definition of persistent views over different databases (Litwin et al .90). These systems guarantee that the appropriate connections are properly established as defined by the view. Therefore they allow access to distant data, but do not support any automatic enforcement of consistency constraints among the various databases.

The research scene currently focuses on federated database (FDB) systems (Sheth 1990). FDB systems aim at scalable integration, combining data integration and site autonomy requirements. They allow each database administrator to define the subset of the local data, if any, which is to be made available to users of the federated system. These subsets are integrated into one (or more) virtual DB, called the FDB. *Virtual* here refers to the fact that only the schema of the FDB is created. Instances of the FDB have no materialized existence. They are temporarily created on the fly according to user requests. Integration, as well as import/export of data into/from the FDB, is managed by the federated system, possibly on the basis of a standard data model and manipulation language.

While gateways and view systems basically provide users with a multidatabase access language (usually, some SQL dialect), without any unification of the semantics of data from the various sources, federated database systems promote an integrated view of the data they manage. Users can therefore access the FDB like a centralized database, without having to worry about the actual physical location of data or the type of the local DBMS. This explains why the federated approach is so popular today. However, before FDB systems come to the market, a number of issues have to be solved. These include design issues, related to the establishment and representation of a common understanding of shared data, as well as operational issues, related to adapting database or GIS techniques to the new challenges of distributed environments. The former focus on database integration, cooperative work, schema/DB evolution, while the latter investigate system interoperability mainly in terms of supporting exchange of objects and methods, new transaction types (long transactions, nested transactions,...), new query processing algorithms, security rules, open system architectures, and so on.

The kernel of design issues is the database integration problem. No surprise therefore that a large number of papers have investigated various facets of database integration. An overview of the existing approaches and of remaining open issues for non spatial databases is presented in (Parent and Spaccapietra 1998). This paper focuses on integration issues specific to spatial databases. This is an area where integration of existing data is currently done by hand, using important manpower resources. By dramatically reducing the cost of this process, database integration techniques are expected to play a major role in promoting new uses of existing data.

The next section briefly outlines the database integration process. A detailed presentation may be found in (Spaccapietra et al. 1992). Section 3 introduces the example that will be used throughout the paper to illustrate our proposal. Section 4 analyzes what information is needed to precisely identify and describe the inter-schema correspondences

which will direct the integration process when spatial data is involved. Section 5 discusses in some detail one of the most frequent conflict opposing diverging representations of related data from different spatial databases. Section 6 shows the result of integration on the example. Finally, the conclusion points at the need for further work on some major issues in spatial data management.

2. Integration methodology

In the GIS domain it is quite common to find several data sources describing, at least partly, the same geographical space. Usually, data are collected for specific purposes, very different from one source to the next one. As no strong guidelines exist for data collection, the existing sources rarely describe the same space in the same way. Therefore, when the decision is taken to integrate various data sources into a single framework, the major problems are: 1) developing a correct understanding of the semantics of existing data, i.e. what they really mean, 2) establishing an accurate correlation structure to avoid comparing apples and oranges, and 3) choosing a well-suited integrated description, based on integration goals and on the available data conversion techniques. These many facets are taken into account by organizing database integration methodologies into a three phases process:

1 **schemas preparation**: this phase includes all preparatory activities which aim at the convergence of existing descriptions towards a uniform pattern. Examples of such activities are: transforming all descriptions to fit a unique format (in database terms, translating schemas in heterogeneous data models into equivalent schemas expressed in a common data model; in GIS databases, this may include turning raster data into vector data, or viceversa), enriching and complementing existing descriptions with additional descriptive information to achieve a uniform level of understanding of data semantics (e.g. adding an explicit definition of the reference system to make sure that coordinate values for geographic data are properly interpreted), establishing global and local thesauri to ease information exchange.

2 **correspondences investigation** (including detecting the conflicts): this phase aims at the identification and precise description of all correlations among existing descriptions (inter-schema correspondences, at the meta level) and among existing data objects (inter-database correspondences, at the data level). Prototype tools exist to analyze similarities among schema elements (Hayne and Ram 1990). However, the final decision on whether a correspondence holds or not is the responsibility of database administrators, based on their knowledge of the semantics of data.

3 **integration**: this phase aims at solving the possible conflicts, and creating the integrated description as a virtual schema on top of the existing data sources. Again, the database administrators take part into the process to define the criteria for building the integrated description. Alternative criteria include: minimality (to make the result as simple as possible by reducing the number of data types), exhaustiveness (to ease discussions with local database administrator by including into the result every data type from the existing descriptions), enforcement of given data modeling standards (in particular, to apply some administrative policy on data presentation).

Many methodologies have been proposed in the traditional database literature to cover a specific phase or the whole spectrum in the database integration process. Readers are referred to (Parent and Spaccapietra 1998) for a survey of the problems and of the solutions. As for spatial databases, a first sketch for a global framework to implement GIS interoperability is reported in (Laurini 1998). Other works typically develop specific techniques to address some of the issues in GIS data conversion. Examples are transformations between raster and vector data (Piwowar et al. 1990), coordinates transformation (Bugayevskiy and Snyder 1995), geometry matching using overlay techniques (Frank 1987) (Dougenik 1980) (Pullar 1993) (Demirkesen and Schaffrin 1996) or rubber-sheeting techniques (Fagan and Soehngen 1987) (Laurini 1994), surface aggregation (Flowerdew 1992), and automated generalization in multiscale databases (Ruas and Plazanet 1996).

The basic split among existing approaches is between: 1) manual approaches, i.e. those methodologies which aim at providing database administrators with adequate tools to perform integration; basically, such tools are schema definition or manipulation languages, and 2) semi-automatic approaches, where the aim is to have integration automatically performed by a tool based on the correspondence assertions given by the database administrator. We believe that the second approach, semi-automatic integration, better suits the needs of GIS applications: GIS data administrators are not necessarily computer science specialists, while their application experience in the semantics of geographic data gives them an ideal role to interact with a CASE tool for the specification of correspondences. The rest of the paper assumes the semi-automatic integration approach.

3. A running example

An example inspired from the cartographic and topographic databases maintained at the French National Geographic Institute will serve throughout the paper to illustrate the integration problems. The example shows records for two road network databases describing the same geographical area. The first database, DB1, is roughly equivalent to a 1:250'000 scale. Database DB2 is more detailed and is roughly equivalent to a 1:10'000 scale. The real world phenomena are not represented with the same degree of detail within the smaller scale. Notice that, strictly speaking, the notion of scale (the ratio between the size of an object on the map and its real size on the ground) is a cartographic concept and does not apply to abstract representations stored in spatial databases. The appropriate concepts are: **precision** (degree of detail in the reporting of a measurement), **accuracy** (relationship between a measurement and the reality which it purports to represent) and **resolution** (the smallest object which can be represented) (Goodchild 1991, Müller et al. 1995).

The schema of database DB1, illustrated in figure 1, contains the following object types:

- **RoadSection** describes the basic components of the road network as seen by this database. A road section is a piece of road, homogeneous in value with respect to the set of attributes and relationships. For example, if the number of lanes changes along the road, each change generates a split into a new road section. Road sections are

described by the number of the road they belong to and the number of lanes in each directions. Road sections have linear geometry.

- **Node** describes points in the road network where traffic conditions change, thus delimiting road sections. A node is the begin- or the end-extremity of a road section. A node can be a crossroad, a traffic circle, a toll, a dead end, or a point where the value of some attribute of the road changes. This is described by the *kind_of_node* attribute. A node can delimit several road sections, each road section has one and only one begin node and one and only one end node. Nodes have point geometry.
- **Overstepping** describes points where several road sections at different levels pass on/under each other with no connection. An overstepping may consist of one or several overlapping bridges, but this is not recorded. There is a nxn association, named *on/under*, between oversteppings and road sections (each overstepping is linked to at least two road sections, a road section may be linked to 0 or any number of oversteppings). Each link records the level in the overstepping of the associated road section (0 if the road section runs on the ground level, 1 if it runs on the first bridge, etc.) and the height of the bridge. Oversteppings have point geometry.

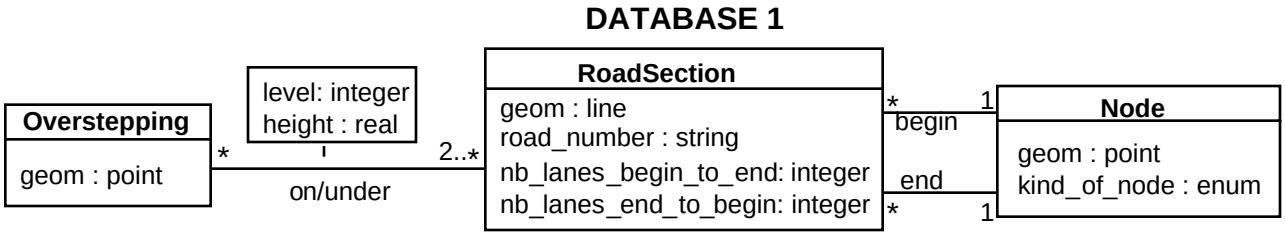


Figure 1: the schema diagram for database DB1 (in the UML notation (Booch et al. 1997))

The schema of database DB2, illustrated in figure 2, contains the following object types:

- **WaySection** is the basic part of the road network. It is composed by one or several contiguos lane(s) going in the same direction (from the beginning of the way section to its end). If the road has several lanes in the same direction split by a separator (for example, one bus lane is physically separated from two car lanes) then each group of lanes is represented as a separate way section. A way section is homogeneous in values for the set of its attributes and relationships. A way section is described by the number of the road it belongs to, the number of lanes it has and its position (either "surface" or "tunnel", the latter if it runs inside a tunnel). A way section has linear geometry. Its end points are known through the *begin* and *end* links with Extremity.
- **Separator** describes separators between two way sections of the same road. A separator may separate sections going in the same direction, as in the case of a bus way section and a car way section, or way sections going in opposite directions, as it is the case for highways. A separator has linear geometry and is characterized by its width. It is linked to the way sections it separates.

- **Extremity** is a point or an area where traffic conditions change. An extremity is either a crossroad, a traffic circle, a toll, a dead end, a value-change node or an end of tunnel. Only the **Crossroad**, **Toll** and **EndofTunnel** subtypes are materialized in the schema. Because these subtypes have different geometries, the geometry of Extremity is geo (where "geo" denotes any geometry).
- **Bridge** is a surface structure which allows several way sections to intersect with no connection. A bridge has a given name (*toponym*) and has surface geometry.

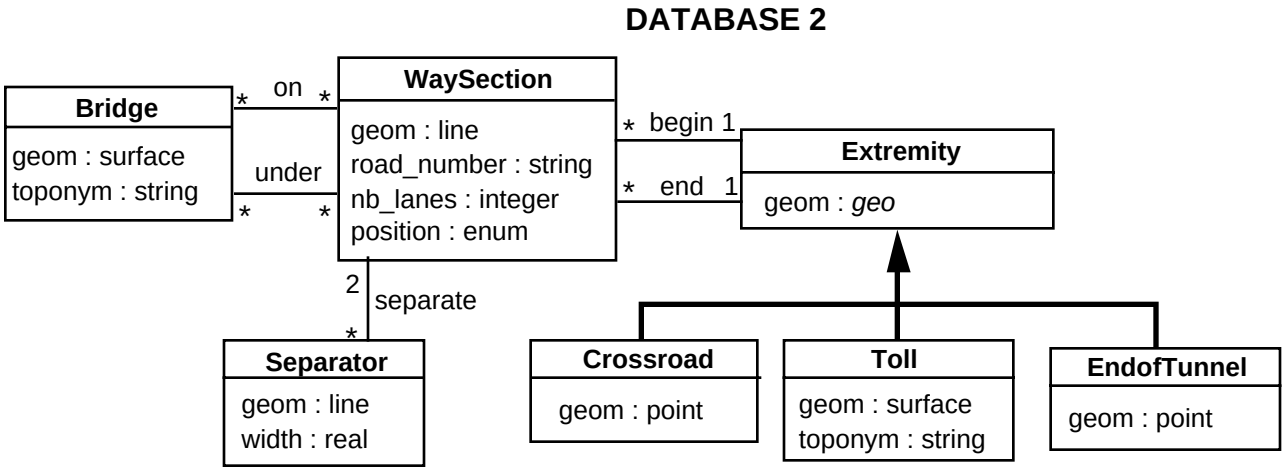


Figure 2: the schema diagram for database DB2

4. Inter-schema correspondences

A prerequisite to database integration is the precise identification of inter-schema relationships (phase 2 in section 2). Integration implies that whenever the existing databases contain duplicate, complementary or otherwise related descriptions of the same real world phenomena, these descriptions should be appropriately merged to provide a single picture of the overall data. This kind of merging is performed at the type level, resulting in the integrated schema, and virtually at the instance level, resulting in the integrated database.

The definition of an inter-schema relationship is called an **inter-schema correspondence assertion** (ICA). Assuming S_i and S_j denote the schemas of two databases to be integrated, the general format for an ICA includes four clauses:

S_i.item$_i$ *correspondence-relationship* S_j.item$_j$
SDM instance-matching-predicate
WCA attribute-correspondences
WCG geometry-correspondence

where $item_i$ and $item_j$ are names of a schema element (or expressions involving schema elements) in S_i and S_j, respectively, and *correspondence-relationship* is one of the usual set relationships: $\equiv, \subset, \subseteq, \cap, \supset, \supseteq, \neq$.

These four ICA clauses are detailed in the following subsections.

4.1 What are the related schema elements ?

In the simplest case, instances of a given type (e.g. $item_i$) in one database are one to one related to instances of a given type (e.g. $item_j$) in the other database. In this case, related elements are just denoted by their qualified names:

$S_i.item_i \equiv S_j.item_j$

The equivalence states that the set of real world things represented as $item_i$ instances in database S_i is, at any time, the same as the set of real world things represented as $item_j$ instances in database S_j. The 1:1 mapping holds independently of which data structures or values actually represent that set in the two databases.

More frequent is the case where related set of real world things are not exactly equivalent. The most generic case is when they intersect. It may also be that one includes (is more generic than) the other one. For instance, in our running example the S1 type *Node* is included into the S2 type *Extremity*: in addition to crossroads, traffic circles, tolls, dead ends, and value-change points represented in *Node*, *Extremity* also represents end-of-tunnel points. Moreover, due to the different resolutions of the two databases, *Node* only stores a subset of the nodes stored by *Extremity*. This may be described by the ICA:

S1.Node $\subset$ S2.Extremity

However, this ICA can be made more precise by observing that instances of *Node* can be grouped according to the value of the *kind_of_node* attribute, so that a group corresponds to a specific subtype of *Extremity*. This may be described, for example, by the ICA:

SELECTION S1.Node (*kind_of_node* = "crossroad") $\subset$ S2.Crossroad

where SELECTION is the usual algebraic operator which builds the subset of instances which obey a given predicate. Any algebraic expression (with selections, union, joins, ...) may be used as needed to specify the set of instances involved in the correspondence (Dupont 1994).

The above examples were discussed as 1:1 or 0:1 mappings between populations (i.e. sets of instances) of object classes in two different databases. More precisely, because of the difference in resolution in our example, one node of database S1 may correspond to several extremities in database S2. Therefore this particular mapping is 1:n, but in the sense that for one node instance there may be several extremity instances such that each one corresponds to the given node. More generic cases are those involving a 1:n or m:n mappings at either the schema or the instance levels. We term fragmentation/aggregation conflicts those 1:n or m:n mappings at the instance level where the n instances represent the fragments to be combined together to play the role of the corresponding thing in the ICA. These conflicts arise when some real world phenomenon can be perceived differently by different designers, so that one designer sees it as a whole (one thing) while another designer sees it as a group of interrelated things of different types (the fragments). To indicate that several elements are the fragments involved in a correspondence with the aggregated object, we use a SET expression. For example, the following ICA expresses the fact that an instance of *RoadSection* in S1 describes the same real world thing as described in S2 by a group of several instances of *WaySection* and of *Separator*.

S1.RoadSection $\subseteq$ S2.SET (WaySection, Separator)

Section 5 will discuss in more detail and precision these more complex correspondences.

Objects and their attributes are not the only elements forming a schema. Links also contribute to the description of real world phenomena, including composition links, association links, is-a (generalization/specialization) links, etc. Inter-schema correspondences between links (and, more generally, between paths resulting from composition of links) have also to be identified and described to achieve a correct integration of the input schemas. For sake of simplicity as for lack of space we do not discuss link integration in this paper. Interested readers may find a discussion of link integration in (Spaccapietra et al. 1992).

4.2 How are their populations related ?

As already informally seen in the previous subsection, an ICA has to specify which set relationship holds between the sets of real world things designated in the correspondence. An equivalence assertion states that at any time the two sets are the same. An inclusion assertion states that at any time one set includes the other one. An intersection assertion states that at some time the two sets may have some common elements. Finally, a disjointedness assertion states that at any time the two sets do not have common elements, despite the fact that they may be regarded as semantically related. Such an assertion is useful when an integration at the integrated schema level is desirable, although the respective populations are disjoint. It is worthwhile noting that the equivalence and inclusion cases induce strong synchronization constraints on insert/delete operations on the related databases. Knowing which set relationship holds is essential in order to determine how to accurately integrate the related elements.

4.3 How are corresponding instances identified ?

In order for an integration to be operational, i.e. to allow access to interrelated objects via the integrated schema, the federated system has to know how to find in the local databases the different representations of the same real world thing. Only if some logical link among these representations is available, all the information can be gathered together for the benefit of users of the federation.

To instruct the system on which data materializes such logical links, each ICA has to specify the mapping between the corresponding instances. Most often, at least in non spatial databases, value-based identifiers (e.g. primary keys in relational models) can be used to map corresponding instances to each other. This, however, does not have to be the case, and any 1:1 mapping function is acceptable. Spatial databases offer a specific alternative for identification of correlated objects: by location, i.e. through their position in space. This allows to assert that two instances are related if they are located in the same point (or area, or volume) in space. In our running example, all correspondences between related object classes, e.g. *Node* and *Extremity*, need an instance mapping mechanism based on the geographic position, simply because there is no thematic attribute to serve as an object identifier.

The SDM (for Spatial Data Matching) clause in ICAs specifies the instance matching predicate which defines the correspondence at the instance level. The predicate may be a simple correspondence between identifying attributes, a simple correspondence based on

equality of coordinates or on identical topological position (Lemarié and Raynal 1996), (Devogele et al. 1996), or a complex correspondence computed through standard functions or ad-hoc methods, which may operate on both thematic and spatial data. For instance the above ICA on crossroads could be extended to:

SELECTION S1.Node (*kind_of_node* = "crossroad") ⊂ S2.Crossroad
SDM S2.Crossroad INSIDE BUFFER (S1.Node, resolutionS2)

where INSIDE is the usual topological relationship which in this example checks if the point locating the crossroad from S2 is within the area computed by the BUFFER function, which transforms the point representing the S1 node into a surface geometry according to the resolution of the S2 database (Devogele et al. 1996).

A similar SDM holds for tolls:

SELECTION S1.Node (*kind_of_node* = "toll") ⊂ S2.Toll
SDM S1.Node INSIDE S2.Toll

Because S2.Toll geometry is a surface, we simply use INSIDE to express the fact that the point representing the Node must be in the surface representing the toll.

More examples will be discussed in the next section.

4.4 How are representations related ?

The last specifications needed to complete an ICA deal with correspondences between representations of related elements, in terms of either their thematic attributes or their geometry. Knowing that the related types comprise one or more common attributes (even if they are coded differently) allows to avoid duplication in the integrated schema. This attribute correspondence is expressed by a WCA (With Corresponding Attributes) clause. A WCG (With Corresponding Geometry) clause is used whenever it is possible to specify how the geometry in one database is derivable from the geometry of the related element in the other database.

For example, the ICA relating S1.RoadSection to S2.WaySection includes the WCA clause:

WCA road_number = road_number,
nb_lanes_begin_to_end = f(nb_lanes),
nb_lanes_end_to_begin = g(nb_lanes)

This states that the value of the *road_number* attribute in a RoadSection instance is the same as the value of the *road_number* attribute in all WaySection instances corresponding to the RoadSection instance. It also states that RoadSection values for the *nb_lanes_begin_to_end* and *nb_lanes_end_to_begin* attributes are derivable from the values of the *nb_lanes* attribute in the corresponding WaySection instances. The derivation is through the *f* and *g* methods. These will be explained in the next section.

The WCG clause for the same ICA would be:

WCG RoadSection.geometry =
MERGE UNSPLIT ({x.geometry /x∈ SET (WaySection, Separator) })

This states that the geometry of a given RoadSection instance may be computed from the geometries of the corresponding WaySection and Separator instances through the UNSPLIT of their lines, which groups adjacent line segments into a continuous line, and the MERGE of these continuous, quasi-parallel lines into a single line.

A set of ICAs for the running example is given separately in the appendix.

5. Solving fragmentation / aggregation conflicts

As stated above, we use the term "fragmentation/aggregation conflict" to denote those situations where there is either a 1:n (n>1) or a n:m a (n>1, m>1) correspondence between instances from two databases, such that the same real world thing is represented on one side as one instance (or a set of fragment instances) and on the other side as many fragment instances. More precisely, we say that there is a n:m correspondence when there does not exist any simpler 1:1 or 1:n mapping between the corresponding instances. Fragmentation conveys the image that one element is cut into several pieces (the fragments), while aggregation refers to the inverse process of forming a single (aggregated) element from a set of elements. These conflicts are frequent in spatial databases. In particular, they are likely to occur whenever the related databases have different resolutions, as in our running example. For instance, each road section of DB1 may correspond to several way sections and separators of DB2, as illustrated in figure 3. Similarly, each node of type traffic circle of DB1 may correspond to several connected crossroads and way sections of DB2. These correspondences are 1:n correspondences at both the type and the instance levels.

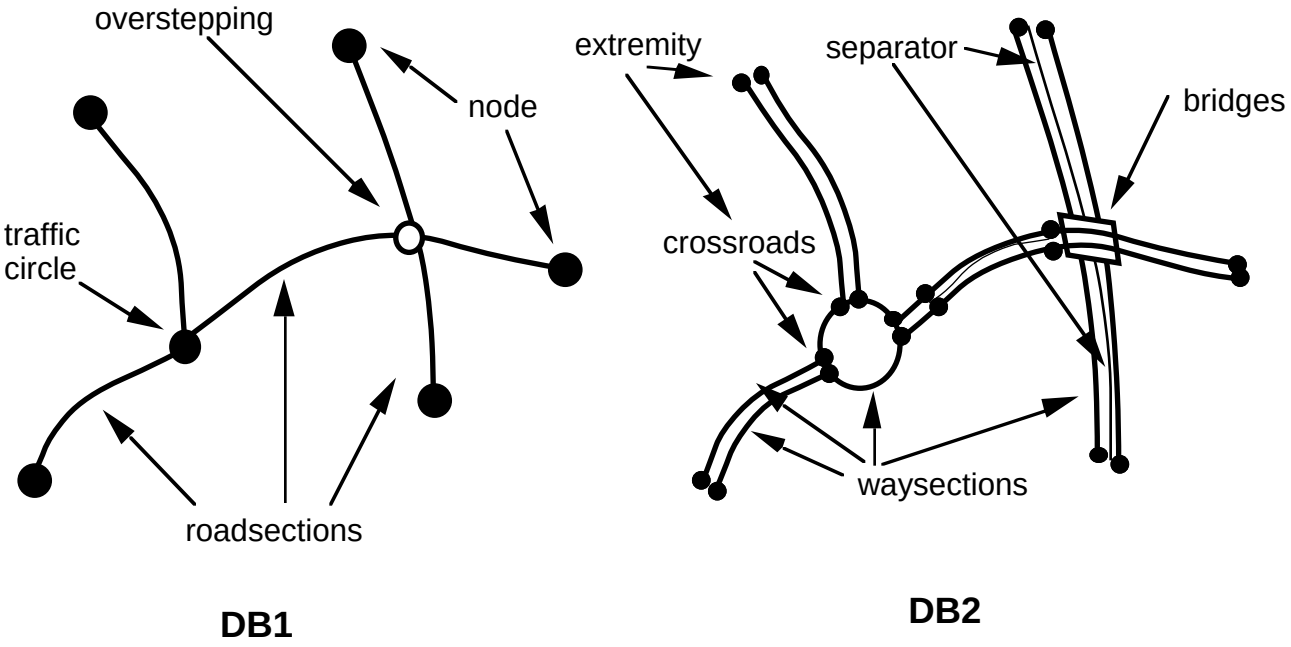


Figure 3: different viewpoints on the same road network

This type of complex conflict may be solved using the data modeling concept known as aggregation (also called composition link or part-of relationship). This concept relates an object to its component objects. According to (Booch et al. 1997) recommendations, we graphically denote an aggregation with a small diamond close to the aggregated object class, as illustrated in figure 4.

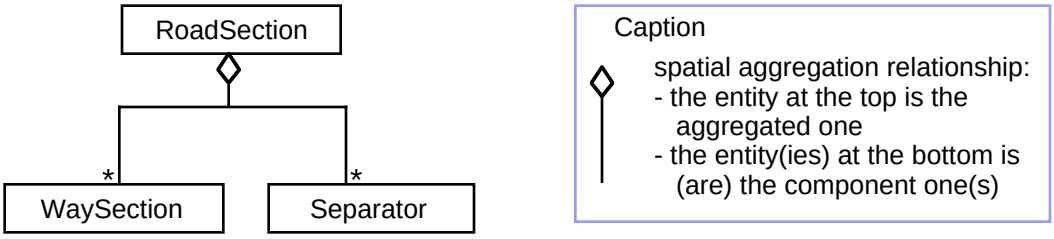


Figure 4: spatial aggregation relationship

The specification of an ICA involving an aggregation/fragmentation conflict such that one instance of some type X in schema S1 corresponds to a set of fragment instances of some type Y in schema S2 may be seen as a two steps process, as illustrated in figure 5:

- the first step consists in the definition of a virtual type Y' in S2, to represent the virtual instances resulting from the aggregation into a single element of the Y fragments corresponding to one instance of X;
- the second step consists in stating the ICA between X and Y'. The SDM clause in the original ICA provides the criteria for the aggregation of Y instances into virtual Y' instances.

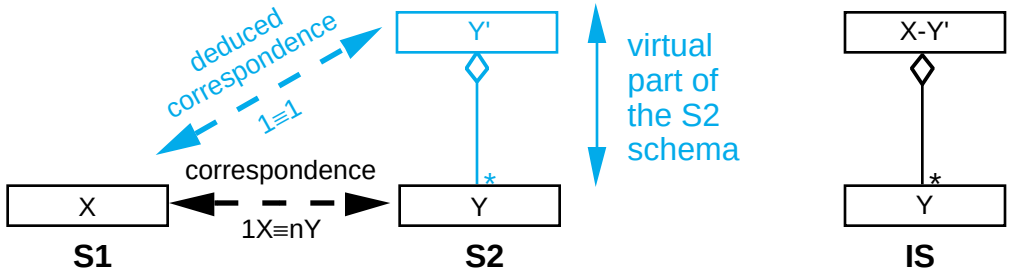


Figure 5: solving 1:n fragmentation/aggregation conflicts

Let us consider the 1:n correspondence between DB1 road sections (playing the X role in figure 5) and DB2 way sections and separator instances (playing the Y role). The virtual Y' type is specified using an aggregation operator that we denote here as SET. Knowing that a road section matches with 1 or more way sections and possibly one or more separators, the virtual type is defined as:

SET ([1:N] WaySection, [0:N] Separator)

Elements of this type are multi-sorted sets, where the numbers in brackets specify the minimum-maximum cardinalities of each set. Each instance of this type has to obey these cardinalities.

To specify how instances of the virtual type are built from instances of the component types we need in this particular example to state the following constraints: 1/ in any one instance of the virtual type its composing *WaySection* instances must belong to the same road (i.e. they have identical value for *road_number*) and be "next" to each other (either parallel or in a sequence forming a line), and 2/ in any one instance of the virtual type its composing *Separator* instances, if any, must be linked to the composing *WaySection* instances via the *separate* relationship. These constraints may be expressed as a complex predicate in a query language. For sake of simplicity, functions specific to spatial data management may be used. In this case, usual query expressions are used to assert that

way sections and separators belong to the same road (thematic predicates), while spatial functions are used to convey the other constraints and check them against the spatial types and geometries of way sections, separators and road sections. The ICA for road sections may be stated as:

S1.RoadSection ⊆ S2.SET ([1:N] WaySection, [0:N] Separator)
SDM SET ([1:N]WaySection) = { w/ w∈WaySection ∧
w.road_number = RoadSection.road_number ∧
w.geometry INSIDE BUFFER (RoadSection, resolutionS2) }
SET ([0:N]Separator) = { s/ s∈Separator ∧
s.geometry INSIDE BUFFER (RoadSection, resolutionS2) ∧
∃w∈ SET([1:N]WaySection) (separate(s,w)) }

The first clause specifies that every road section instance corresponds to a multi-sorted set of DB2 instances. The first predicate in the SDM clause specifies that for each road section instance only way section instances 1/matching the road number, and 2/whose geometry lies within a given buffer surface enclosing the road section geometry should be considered. The second predicate restricts separator instances to those 1/linked to the current set of way section instances, and 2/whose geometry lies within the buffer surface enclosing the road section geometry.

As shown in figure 5, the integrated schema (IS) will include the type Y, its aggregation mechanism and the type resulting from the integration of the X and Y' types, according to integration rules for 0,1:1 correspondences. Figure 4 shows the result of the integration of Road Sections and Way Sections and Separators.

More complex n:m conflicts, where corresponding "instances" on both sides are in fact sets of instances of one or more types, can be dealt with in a similar manner (see figure 6). First, a virtual aggregated type is built for each schema. Second, a 0,1:1 correspondence is defined between the two virtual types. Finally, the two schemas are integrated.

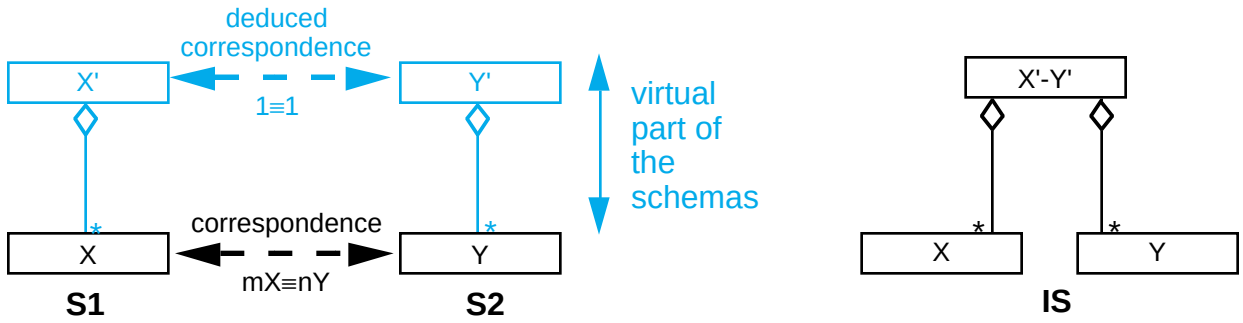


Figure 6: solving n:m fragmentation/aggregation conflicts

The correspondence between *Overstepping* and *Bridge* is such a n:m correspondence. The integrated schema will contain the construct shown figure 7, which depicts two alternate spatial aggregations: each *Set-of-Oversteppings/Bridges* instance can be seen as made up of instances either of *Overstepping* or of *Bridge*.

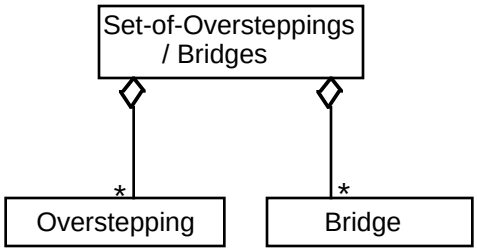


Figure 7: integration of Overstepping and Bridge

As far as attributes are concerned, corresponding attributes can either be defined at the same level of resolution in the two databases, or one attribute can be defined as the result of the aggregation of the values taken by the other attribute in the component entities. For example, corresponding attributes between road sections of DB1 and way sections of DB2, are:

- *road_number*, which has the same value in both databases. It is a "constant" attribute: it has the same value in each of the component entities;
- *nb_lanes* of *WaySection* is a "variable" attribute of DB2: it can have a different value in each component entity. It has to be aggregated to construct the unique value of the attribute *nb_lanes-begin-to-end* of DB1. In this case the aggregated function is not a standard one (like SUM, MIN, MAX, COUNT, AVG); it has to be provided by the user. This aggregation function, *count-lanes*, counts the number of parallel lanes which are headed in one direction.

The WCA clause for the ICA on *RoadSection* is the following:

WCA road_number = road_number ,
nb_lanes_begin_to_end = count_lanes (RoadSection2 , DIRECTION (RoadSection)),
nb_lanes_end_to_begin = count_lanes (RoadSection2, OPPDIRECTION (RoadSection))

where RoadSection2 is the name of the virtual aggregated type built from way sections and separators.

Finally, the geometry of the aggregated entity can correspond to the geometric union of the geometries of the component entities, or to a generalization of it. For example, the correspondence between the geometries of road sections and the geometries of way sections and separators can be:

WCG RoadSection.geometry = MERGE UNSPLIT ({x.geometry / x∈ RoadSection2})

where UNSPLIT is the spatial function which in this case forms a continuous line from a set of connected segments, and MERGE is the spatial function which in this case generalizes way section and separators lines into a single, road section line.

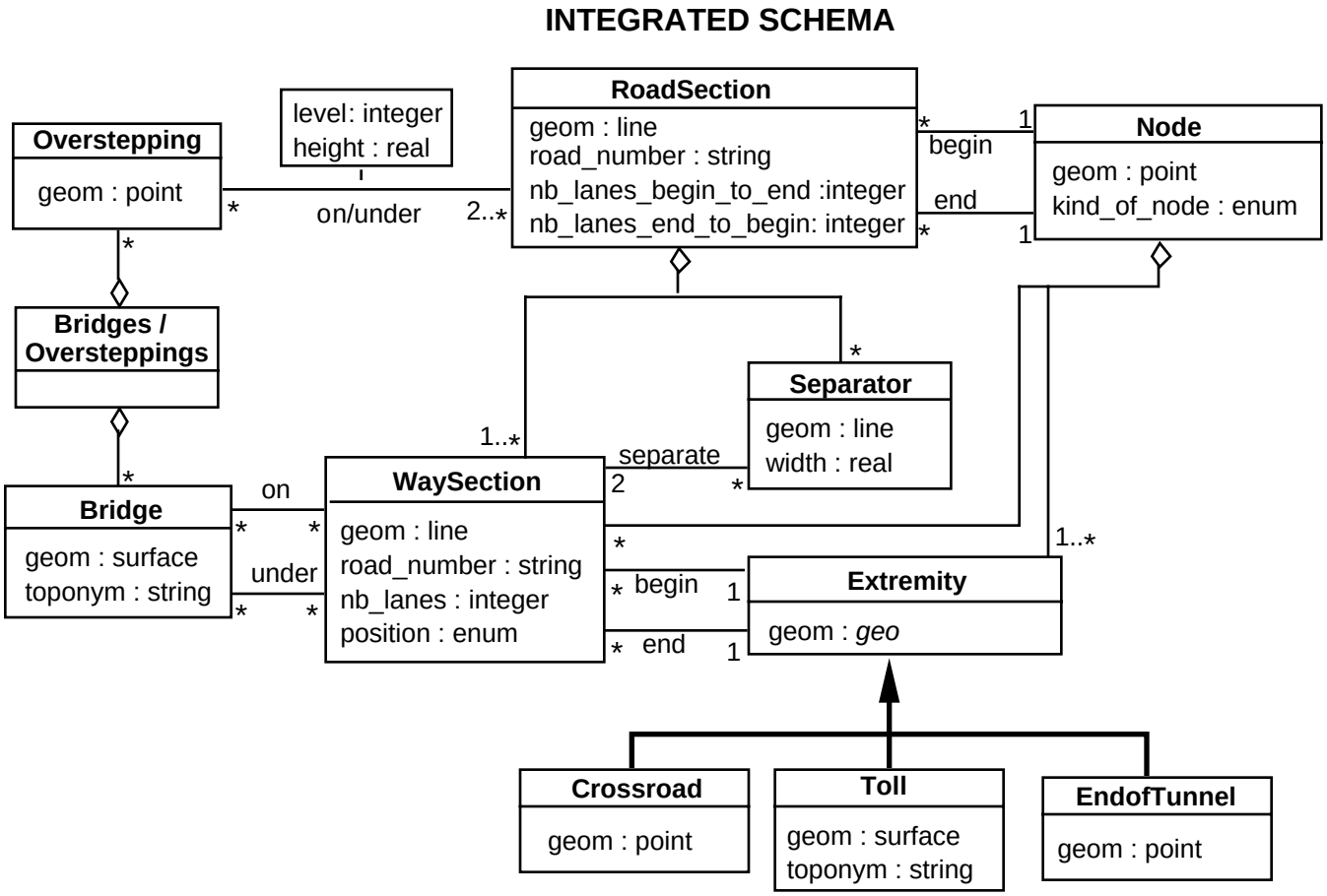
6 The integrated schema

Once the correspondences have been found and precisely described, the integrated schema (IS) can be built. Actually, there can be more than one IS. Given two databases and a set of correspondences, one can build different IS according to the purpose of the future IS.Another usage of correspondences is to derive from their description the methods that

will automatically propagate updates among databases, thus ensuring the permanent consistency of the whole set of databases.

Building the IS is a process which has been described in several papers (Batini et al. 1986), (Nyerges 1989) (Sheth et Larson 1990) (Spaccapietra et al. 1992), and which can be automated to a large extent. For two databases, it can be summarized as:
- for each element (class, attribute or relationship) of the two schemas that has no correspondent in the other schema, put this element into the IS;
- for each correspondence, apply the associated integration rule.

Integration rules have been defined for most simple conflicts (Navathe et al. 1986) (Thieme and Siebes 1993). In the previous section, new integration rules have been defined for fragmentation/ aggregation conflicts. Although some more complex conflicts are still pending, these rules allow us to integrate the two databases of the example. The result is shown in figure 8, where a powerful spatial data model is used. It contains the following concepts:
- spatial and non spatial entity classes,
- non spatial binary relationships,
- inheritance links between entity classes,
- spatial aggregation relationships between spatial entity classes, meaning that one (complex) entity is made up of (component) entities. These aggregation relationships support derivation predicates that specify which component entities are to be aggregated to make up a complex entity, and how thematic attributes and geometry of the complex entity are derived from those of the component entities (and vice versa).



Constraints:
begin(RoadSection, Node) = AGGREGATION(begin(WaySection, Extremity))
end(RoadSection, Node) = AGGREGATION(end(WaySection, Extremity))
on/under(Overstepping, RoadSection) = AGGREGATION(on(Bridge,WaySection), under(Bridge, WaySection))

Figure 8: an integrated schema for the example

In this IS correspondences between paths have not been really integrated: they generated integrity constraints which are equivalent to the path correspondences. Due to the weakness of data models when dealing with relationships, path correspondences can be integrated only in simple cases, as in 0,1:1 correspondences. Integrating 1:n path correspondences would imply to have a data model supporting concepts such as aggregation-relationships between relationships.

7. Conclusion

This paper has provided an overview of the database integration process as it may apply to spatial databases. First, we have shown what the promises of database integration are, and how the integration process can be organized into a sequence of dedicated steps so that the overall complexity becomes manageable. Next, we have introduced an example drawn on a concrete application currently in use at the National Geographic Institute in France. With the support of the example, we have discussed how inter-schema correspondences may be described. They serve as input to the semi-automatic integration phase, where conflicts arising from differences in information representation are solved and an integrated schema, describing all available data, is built.

We have devoted a more detailed analysis to the fragmentation/aggregation conflicts, where correspondences are based on a 1:n or n:m mappings between instances. These conflicts frequently arise when related spatial databases have been set up based on different scales. We are not aware of any prior work on these conflicts.

Our analysis has shown that full integration of spatial databases, possibly based on different scales, requires a powerful data model for the integrated schema in order not to loose the semantics of the original schemas. This federated data model should include capabilities for the description and management of complex objects, with temporal and spatial features orthogonal to the data structure, generic relationships as well as topological and aggregation relationships, and multiple representations of objects (possibly with different geometries) according to different viewpoints and/or different scales.

We have currently specified such a data model. It is named MADS for Modeling Application Data with Spatio-temporal features (Parent et al. 1997). While MADS concepts are formally defined, its goals are very pragmatic, focusing on user-orientation (each user must be able to understand and interact with the global conceptual schema as well as with the local GIS schema) and on implementability (all MADS features should be supported using appropriate mappings onto the functionalities provided by current GIS, e.g. ArcInfo). A visual interface layer is also being implemented, to include a schema definition editor and a direct manipulation query editor.

APPENDIX

A set of ICAs for the running example (the left side of the correspondences refers to schema S1, the right side to schema S2):

Nodes:

Node ⊆ SET([1:N]Extremity, [0:N]WaySection)

SDM SET([1:N]Extremity) = { e/ e ∈ Extremity ∧
(kind_of_node="crossroad" ⇒ e∈ Crossroad) ∧
(kind_of_node="traffic circle" ⇒ e∈ Crossroad) ∧
(kind_of_node="toll" ⇒ e∈ Toll) ∧
e INSIDE BUFFER (Node, resolutionS2) }

SET([0:N]WaySection) = { w/ w ∈ WaySection ∧
∃ e ∈ SET([1:N]Extremity (begin(w,e)) ∧
∃ e ∈ SET([1:N]Extremity (end(w,e)) }

WCG S1.Node.geometry =
GENERALIZATION (S2.SET([1:N]Extremity, [0:N]WaySection)).geometry)

Road sections:

S1.RoadSection ⊆ RoadSection2: S2.SET ([1:N] WaySection, [0:N] Separator)

SDM SET([1:N]WaySection) = { w/ w ∈ WaySection ∧
w.road_number = RoadSection.road_number ∧
w INSIDE BUFFER (RoadSection, resolutionS2) }

SET([0:N]Separator) = { s/ s ∈ Separator ∧
s INSIDE BUFFER (RoadSection, resolutionS2) ∧
∃ w ∈ SET([1:N]WaySection (separate(s,w)) }

WCA road_number = road_number ,
nb_lanes_begin_to_end = count-lanes (RoadSection2 , DIRECTION(RoadSection)),
nb_lanes_end_to_begin = count-lanes (RoadSection2, OPPDIRECTION(RoadSection))

WCG RoadSection.geometry = MERGE UNSPLIT({x.geometry / x∈RoadSection2})

RoadSection—begin—Node ⊆ Roadsection2—begin—Extremity

RoadSection—end—Node ⊆ Roadsection2—end—Extremity

Oversteppings and bridges:

SET(Overstepping) ⊆ SET(Bridge)

SDM SET(Bridge) = { b/ b ∈ Bridge } such that:
∀ b1∈ SET(Bridge) ∃ b2 ∈ SET(Bridge) (INTERSECT (b1, b2))

SET(Overstepping) = { s/ s ∈ Overstepping ∧ ∃ b ∈ SET(Bridge) (s INSIDE b) }

RoadSection—on/under—Overstepping ≡ RoadSection2—on—Bridge

RoadSection—on/under—Overstepping ≡ RoadSection2—under—Bridge

References

BATINI, C., LENZERINI, M., and NAVATHE, S.B., 1986, A comparative analysis of methodologies for schema integration. In ACM Computing Surveys, 15 (4) , pp. 323-363.

BOOCH, G., RUMBAUGH, J., and JACOBSON, I., 1997, Unified Modeling Language User Guide (Addison-Wesley).

BUGAYEVSKIY, L.M., and SNYDER, J.P., 1995, Map Projections - A Reference Manual (Taylor & Francis).

CEN/TC 287, July 1996, Geographic Information - Reference Model, European Prestandard, Final Draft prENV 12009.

DEMIRKESEN, A., and SCHAFFRIN, B., 1996, Map Conflation : Spatial point data merging and transformation. In Proceedings of GIS/LIS'96, pp. 393-404.

DEVOGELE, T., TREVISAN, J., and RAYNAL, L., 1996, Building a Multi-Scale Database with Scale-Transition Relationships. In Proceedings of 7th International Symposium on Spatial Data Handling, Delft, The Netherlands, M. Moleenar and M-J. Kraak (Eds.), Taylor & Francis, pp. 337-351.

DOUGENIK, J., 1980, WHIRLPOOL : A geometric processor for polygon coverage data. In Proceedings of Auto-Carto 4 (ACSM-ASPRS), pp. 304-311.

DUPONT, Y., 1994, Resolving fragmentation conflicts in schema integration. In Proceedings of Entity-Relationship Approach - ER'94, P. Loucopoulos (Ed.), LNCS 881 (Springer-Verlag), pp. 513-532.

FAGAN, G., and SOEHNGEN, H., 1987, Improvement of GBF/DIME File Coordinates in a Geobased Information System by Various Transformation Methods and Rubbersheeting Based on Triangulation. In Proceedings of Auto-Carto 7 (ACSM-ASPRS), Baltimore, pp. 481-491.

FLOWERDEW, R., 1992, Spatial Data Integration, in Geographic Information Systems, edited by D. J. Maguire, M. F. Goodchild, D. W. Rhind (London : Longman), pp. 337-358.

FRANK, A., 1987, Overlay processing in spatial information systems. In Proceedings of Auto-Carto 8 (ACSM-ASPRS), Baltimore.

FREW, J., CARVER, L., FISCHER, C., GOODCHILD, M., LARSGAARD, M., SMITH T., and ZHENG, Q., 1995, The Alexandria Rapid Prototype: building a digital library for spatial information. In Proceedings of ESRI User Conference.

GEO2DIS, 1997, GEO2DIS: a Client-Server architecture, on Internet, to document and access geodata stored on heterogeneous GISs, http://www.pisa.intecs.it/projects/GEO2DIS/.

GOODCHILD, M., 1991, Issue of quality and incertainty. In Advances in Cartography, Edited by Müller, pp. 113-139.

HAYNE, S., and RAM, S., 1990, Multi-User View Integration System (MUVIS): An Expert System for View Integration, In Proceedings of IEEE 6th International Conference on Data Engineering, Los Angeles, IEEE Press, pp. 402-409.

ILLERT, A., and WILSKI, I., 1995, Integration and Randomization of Contributions to a European Dataset. In Proceedings of 17th International Cartographic Conference, 1, Barcelona, pp. 805-812.

INTERLIS, 1991, a data exchange mechanism for land-information systems, Eidgenössiche Vermessungs-direktion, Kompetenzzentrum Interlis/AVS, 29 pages.

ISO/TC 211/WG 1, 1996, ISO 15046-3 Conceptual Schema Language (Working Draft 1.0), ISO/TC 211 N 222 Document, 31 pages.

LAURINI, R., 1994, Sharing Geographic Information in Distributed Databases, In Proceedings of 32nd Annual URISA Conference, Milwaukee, pp. 441-455.

LAURINI, R., 1998, Spatial Multidatabase Topological Continuity and Indexing: a Step towards Seamless GIS Data Interoperability. In International Journal of Geographical Information Systems, this issue.

LEMARIE, C., and RAYNAL, L., 1996, Geographic data matching: first investigations for a generic tool. In Proceedings of GIS/LIS'96, Denver, Colorado, pp. 405-420.

LITWIN, W., MARK, L. and ROUSSOPOULOS, N., 1990, Interoperability of Multiple Autonomous Databases. In ACM Computing Surveys, 22 (3), pp. 267-293.

MÜLLER, J.C., LAGRANGE, J.P., WEIBEL, R., and SALGE, F., 1995, Generalization: state of the art and issues, In GIS and Generalization, edited by J.C. Müller, J.P. Lagrange, R. Weibel (Taylor & Francis), pp. 3-17.

NAVATHE, S., ELMASRI, R., and LARSON, J., 1986, Integrating user views in database design, In IEEE Computer, 19 (1), pp. 50-62.

NYERGES, T., 1989, Schema integration analysis for the development of GIS databases, In International Journal of Geographical Information Systems, 3 (2), pp. 153-183.

OGIS, 1998, The OpenGIS® Guide, Introduction to Interoperable Geoprocessing, OpenGIS consortium http://www.opengis.org/techno/guide.htm.

PARENT, C., SPACCAPIETRA, S. and ZIMANYI, E., 1997, In Conceptual Modeling for Federated GIS over the Web. In Information Systems and Technologies for Network Society, Y. Kambayashi et al. (Eds.), World Scientific, pp. 173-182.

PARENT, C., and SPACCAPIETRA, S., 1998, Database integration: an overview of issues and approaches. In Communications of the ACM, to appear.

PIWOWAR, J., LEDREW, E. and DUDYCHA, D., 1990, Integration of spatial data in vector and raster formats in a geographic information system environment. In International Journal of Geographical Information Systems, 4 (4), pp. 429-444.

PULLAR, D., 1993, Consequences of using a tolerance paradigm in spatial overlay. In Proceedings of Auto-Carto 11 (ACSM-ASPRS), pp. 288-296.

RUAS A., and PLAZANET, C., Strategies for Automated Generalization. In Proceedings of 7th International Symposium on Spatial Data Handling, 1 (6), Delft, The Netherlands, Taylor & Francis, 1996.

SHETH, A., and LARSON, J., 1990, Federated database systems for managing distributed, heterogeneous, and autonomous databases. IN ACM Computing Surveys, 22 (3), pp. 183-236.

SPACCAPIETRA, S., PARENT C. and DUPONT, Y., 1992, Model Independent Assertions for Integration of Heterogeneous Schemas. In VLDB Journal, 1 (1), pp. 81-126.

STEPHAN, E., VCKOVSKI, A., and BUCHER, F., 1993, Virtual data set: An approach for the integration of incompatible data. In Auto Carto 11 (ACSM-ASPRS) pp. 93-102.

THIEME, C. and SIEBES, A., 1993, Schema integration in object-oriented databases. In Proceedings of the International Conference on Advanced Information Systems Engineering, LNCS 685 (Springer-Verlag), pp. 54-70.

UITERMARK, H., 1996, The Integration of Geographic Databases. In Proceedings of Second Joint European Conference (IOS Press), pp. 92-95.