Beyond Schema Versioning: A Flexible Model for Spatio-Temporal Schema Selection

John F. Roddick (roddick@cs.flinders.edu.au)
School of Informatics and Engineering, Flinders University of South Australia, PO Box 2100, Adelaide 5001, South Australia, Australia.

Fabio Grandi (fgrandi@deis.unibo.it), Federica Mandreoli (fmandreoli@deis.unibo.it) and Maria Rita Scalas (mrscalas@deis.unibo.it)
Dipartimento di Elettronica, Informatica e Sistemistica, Università degli Studi di Bologna, Viale Risorgimento, 2, I-40136, Bologna, Italy.

Abstract. Schema versioning provides a mechanism for handling change in the structure of database systems and has been investigated widely, both in the context of static and temporal databases. With the growing interest in spatial and spatio-temporal data as well as the mechanisms for holding such data, the spatial context within which data items are formatted also becomes an issue. This paper presents a generalised model that accommodates temporal, spatial and spatio-temporal schema versioning within databases.

Keywords: Schema Evolution, Schema Versioning, Spatio-Temporal Databases, Multi- Spatio-Temporal Data Model, MSTDM.

1. Introduction

The motivation for schema versioning (which can be regarded for some purposes as the application of temporal semantics to the schema of a database) is to provide a mechanism for reflecting changes that occur in the real world within the data model in a data independent fashion. For example, structural changes, such as the addition or deletion of attributes, the splitting or coalescing of relations, the addition, modification or deletion of methods, and so on, should be achievable with the minimum impact to both the data and to the applications. Furthermore, semantic changes, such as a change in the interpretation of an attribute's value (a currency attribute, for example, changing from Italian Lira to Euros), should be able to be recorded without any misinterpretation as a result of the attribute's previous semantics.

Schema versioning also provides a mechanism for parallel versions of a database structure to facilitate alternative structural representations to the same data [16]. A previous survey of the field [22] provides a review of the area and a glossary of temporal database concepts [14] gives the following definitions:

Schema Evolution. *A database system supports schema evolution if it permits modification of the database schema without the loss of extant data. No support for previous schemata is required.*

Schema Versioning. *A database system accommodates schema versioning if it allows the querying of all data, both retrospectively and prospectively, through user-definable version interfaces.*

The development of spatial and, more recently, spatio-temporal databases [1, 11, 12, 17] is an important research topic. Recent research has shown that there are similar problems that could be considered as the spatial equivalent to the problems experienced by time-varying data and changing schemata and for which temporal databases and schema versioning provide some solutions. For example, changes in database structure in time and the ability to version schemata have their spatial analogue in the locations to which a given schema is to apply and the ability to apply different schemata according to where the data are to be applicable. This is apparent, for example, in the differences applicable across land administration authorities (qv. [19]) in which data may be collected according to the accepted local cultural norms, customs and laws.

This paper therefore investigates the lessons learnt from the theory and development of temporally oriented schema versioning and argues that similar techniques, with some enhancement, can be applied to spatial and spatio-temporal databases. In doing so, it is acknowledged that the spatial domain differs in several fundamental ways and solutions appropriate to handling time will have to be carefully re-examined for the spatial context. From a semantic point of view, there is no spatial counterpart of the concept of "evolution", which is strictly connected to changes in time. However, there is a straightforward conceptual analogue of temporal versioning in the spatial domain, as different versions can be applicable to different places rather than times.

On the other hand, in a database perspective, schema evolution can be considered as a particular case of schema versioning, where only the last version is retained. Moreover, in a stand-alone database system supporting schema versions (as well as temporal and/or spatial), "standard" schema evolution techniques are employed when deriving a new schema version from existing ones. The focus of this paper is on the *use* of different schema versions in a generalized spatio-temporal context and not on the management of schema changes in producing versions. Hence schema evolution issues will not be considered here in detail.

The paper is structured as follows. Section 2 reviews temporal schema versioning and provides some background to the model to be presented. Furthermore, it investigates the analogies between the temporal and spatial domains and provides some motivation for spatial schema versioning. Section 3 proposes a generalised spatio-temporal schema selection model. Section 4 then addresses some architectural issues, while Section 5 proposes some query language enhancements. Further discussion and research directions are provided in Section 6.

2. Towards Spatio-Temporal Schema Versioning

2.1. Temporal Schema Versioning Revisited

Research into schema evolution and schema versioning has been undertaken for a number of years and has resulted in a number of useful changes to the way in which schema changes are handled. We discuss here briefly three issues (completed schemas, data conversion and query language design) that relate to our research - however, a comprehensive survey can be found in [22].

Firstly, the concept of a completed schema was developed following the ideas of Clifford and Warren [7] for a completed relation to enable all data, regardless of the time of validity, to be accessed. The concept of a completed schema, which was introduced in [21] and discussed in more detail in [23], is that of an overarching schema through which all data, regardless of time of validity, format *or* semantics, can be retrieved. The completed schema (discussed in more detail later) is defined as the minimal superset of relevant schemata capable of holding all associated data without loss.

Secondly, three principal data conversion mechanisms have been employed in the event of change. Firstly, data items are simply coerced to the new format as in [18]. Secondly, a lazy conversion mechanism can be used in which data are converted only when accessed [28]. Thirdly, data are never physically converted and are always accessed through conversion interfaces [4].

Finally, two query language extensions have been proposed which accommodate schema evolution, SQL/SE [21] and TSQL2 [26]. In the most significant of these, the temporal query language TSQL2, an orthogonal schema-time was added to the existing bi-temporal functionality to allow the specification of a designated schema (specified by date) through which the data items are retrieved. For example, a TSQL2 statement such as that shown in Figure 1 specifies that the Employee data held at 1-Mar-1999 for 1-Apr-1999 is to be retrieved using the schema format extant as at 15-Jan-1999. When not specified, schema-time defaults to the current transaction-time and can be used to effect resilience in compiled programs using embedded TSQL2 by including a `SET SCHEMA DATE <compile-date>` clause instead. Note that the `SET SCHEMA` clause operates at query level and heterogeneous schema-time queries cannot be specified.

```
SET SCHEMA DATE '1/15/1999'

SELECT  Employee_Name
FROM    Employee
WHERE   Employee_Dept = 'CS'
AND     VALID(Employee) OVERLAPS '4/1/1999'
AND     TRANSACTION(Employee) OVERLAPS '3/1/1999'
```

Figure 1. TSQL2 Query with Schema-time reference

In [8], the single schema-time dimension of TSQL2 (which was effectively a transaction-schema-time) was expanded to support both transaction- schema-time (intensional schema change) and valid-schema-time (extensional schema change). This allows schemata to be both pre and post-dated as well as specified in data retrieval. In fact, a number of time dimensions can be identified although in some cases it makes little sense to distinguish between some of them.

- **Valid-time.** The time the event occurred or the fact was true in reality.
- **Transaction-time.** The time the data representing the real world event or fact was recorded in the database.
- **Transaction-schema-time.** The time used to determine the structure and format of the data as stored in the database. In [26] this is simply termed schema-time and arguably provides the most useful versioning ability.
- **Valid-schema-time.** The time used to determine the structure of the real world. To date, this has not been widely used as only data about the real world is actually manipulated. It is, nevertheless, the natural partner to valid-time for data.
- **Registration-schema-time.** The time the current schema was updated.
- **Compile-time.** The time the application was compiled and thus the Transaction-schema-time applicable to any static data structures.

- **Decision-time.** The time the decision was taken to invoke the change, cause an event, and so on.

The most significant in terms of this paper are the first four. The relationships between temporal schema dimensions, as well as between the two temporal data dimensions, can be defined as shown in Figure 2 (which augments the diagram given in [20]).

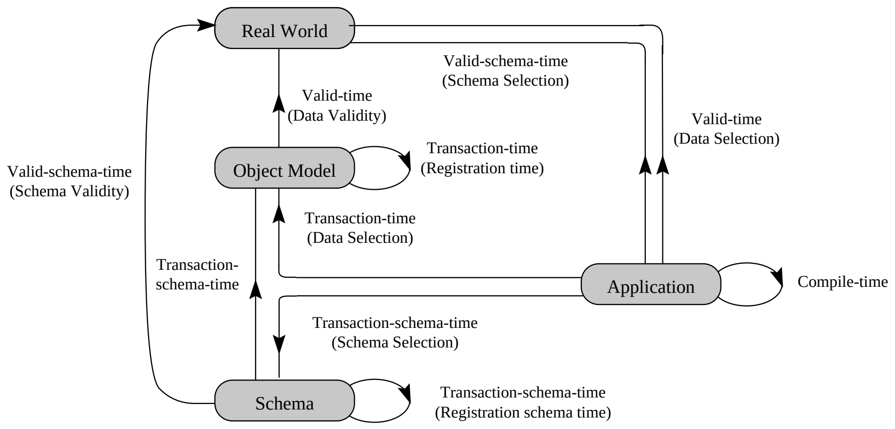

Figure 2. Real World, Object Model and Schema Time Dimensions.

For example, consider the following five statements (D is a data model, E is an event or fact, T_1, T_2, T_3 and T_4 are times and S_1 and S_2 are versions of a database schema):

1. E happened at T_1 (valid-time)

2. E was recorded at T_2 (transaction-time)

3. The structure of D at T_1 (valid-schema-time) was S_1

4. D was recorded at having structure S_2 at T_2 (transaction-schema-time)

5. An application A is compiled (or has a transaction-schema-time set) at T_3. This application may access data stored across multiple valid and transaction-times in a single query.

6. S_1 was recorded at T_4 (registration-schema-time)

Various of these, notably transaction-schema-time, have been discussed in the literature, (see for example [8, 14, 27] and many others). The equivalent schema-time dimensions from the literature, using the terminology used by those publications, are as in Table I although in many

Table I. Schema Dimensions in the Literature.

	Valid-schema-time	Transaction-schema-time
Andany, Leonard and Palisser 1991	$\rightarrow$	$\star$
Ariav 1991	$\rightarrow$	$\star$
Banerjee, et al. 1986	$\rightarrow$	$\star$
De Castro, Grandi and Scalas 1997; Grandi, Mandreoli and Scalas 2000	$\star$	$\star$
Monk and Sommerville 1993	$\star$	$\leftarrow$
Nguyen and Rieu 1989	$\rightarrow$	$\star$
Orlowska and Ewald 1992	$\rightarrow$	$\star$
Roddick and Snodgrass 1995; Roddick 1996	$\rightarrow$	Schema-time

cases, some work has included aspects of multiple dimensions under one heading. Indeed, the most common position in the literature is to assume two or more dimensions of time are the same. Exceptions to this include [3, 8, 10].

In Table I, an arrow indicates that the time specification defaulted to another dimension while a star indicates that no specific notation was applied to the schema time dimension. Note that in some work, it is not clear which time is used and we have had to infer the semantics from the approach taken by the authors, and in some cases the discussion, in their publications.

2.2. Spatial Analogue of Temporal Versioning

It is commonly stated that we perceive a four dimensional world, three dimensions of space and one of time. However, the time dimension has a number of sensory and physical attributes that make simple scaling of a one dimensional solution to four dimensional space-time difficult[1].

- Time is unidirectional and generally, in most systems development, considered as linear. Thus the relational concepts (before, during etc.) are easily understood and accommodated. Space is bi-directional and, particularly in some geographic applications, commonly non-linear.

[1] It is often argued that one of the space dimensions should also be treated differently because of gravity.

- Both time and space are (or can be considered to be) continuous[2], however, it is often far more useful to consider time as discrete and isomorphic with integers, and a larger granularity is often selected. Space, on the other hand, while a specific granularity is sometimes adopted, is often considered as isomorphic with real numbers.

- Common calendar systems, while not universally in operation, are prevalent and a single way of expressing a point in time can usually be adopted for use in an information system. Conversely, multiple sets of rules governing space are not uncommon and frequently two or more systems of spatial coordinates have to be handled within one information system[3].

Notwithstanding these differences, some translations from the temporal to the spatial domain can be made and, particularly with the development of spatio-temporal databases, any possible reuse of accepted conventions would make considerable sense.

Despite the volume of work in schema versioning, including that relating to temporal database systems, schema versioning has not been extended to spatial or spatio-temporal models of data although some of the more general research into schema integration is applicable. Spatial and spatio-temporal databases are becoming popular areas of research and the need for adequate schema versioning problem is becoming evident for these systems also. The added values of spatial schema versioning support would include:

- a mechanism for different schemata (including different constraints) to be applicable according to location. This may be important for jurisdictional, legal or data collection method purposes;

- an indication of the rules under which data items are collected and therefore the manner in which they should be interpreted also in a different place. For example, data collected from one country may have been collected using a different protocol from those collected from another.

- a method for providing local ownership and interpretation of a centrally held database. For example, each authority may provide the rules under which data collected by its authority is interpreted.

[2] Ignoring quantum space-time.

[3] For example, in a map, a road may be represented as a series of connected arcs or as a series of points. The important issue is that conversion between systems may not be lossless.

For example, consider a spatial system of rural maps in which each region has a different categorisation method for each type of entity depicted (towns, agricultural land, natural heritage listing, native title, airport noise contours, flood plains, etc.), as well as perhaps different information provision requirements (authority levels, copyright, freedom of information acts, privacy laws, etc).

Clearly, the task of combining the maps or analysing similar facets of the area necessitates a global schema which needs to combine the elements of the local schemata as determined either by the data itself or by the location to which the data refers. It should be noted that many of the aspects discussed earlier have a spatial analogue. For example, to rephrase the five temporal examples given, consider the following five statements (D is a data model, E is an event, fact or object, L_1, L_2 and L_3 are locations and S is a database schema):

- E happened at L_1 (valid-location)
- E was recorded/observed at L_2 (transaction-location)
- The structure of D for L_1 is S (valid-schema-location)
- D was recorded at having the spatial structure S applicable to location L_2 (transaction-schema-location)
- S was recorded at L_3 (registration-schema-location)

While some of these may not be useful dimensions, it is clear that there is a spatial correspondence which would be useful to investigate further. In this paper, valid-schema-location and transaction-schema-location will be incorporated into a general model of schema selection for spatio-temporal databases.

3. A Model for Spatio-Temporal Schema Selection

In this section, we present our model for the management of spatio-temporal schema versions, which integrates the temporal versioning functionalities with the potentialities of spatial schema versioning.

3.1. An Initial Discussion

In the proposed model, schemata are associated (ie. labelled and referenced) with a given spatio-temporal region (which defaults to "all time and space") plus an optional user-supplied label. When space and time are not supplied the model degrades to a multiple static schema model and when the optional user- supplied label is not supplied we degrade to schema versioning (in space-time). Furthermore, if time only is supplied the model degrades to conventional temporal schema versioning (à la [26]). As we would want, the limit case is degradation to no versioning.

The model embraces the concept of a completed schema discussed in [23] which is constructed as the minimal schema capable of holding all associated data without loss. More precisely, a completed relation scheme C of a relation scheme R contains the minimal union of all explicit attributes which have been defined during the relevant spatio-temporal span of the relation. Moreover, the domain of each attribute in C is syntactically general enough to hold all data stored under every version of R and the implicit primary key of C is defined as the maximal set of key attributes for the scheme over the relevant spatio-temporal span. Versions of the schema can be seen to be views of C. The construction of the completed schema in the event of inter-relational changes can also be accommodated as long as transitional rules between versions are provided. For example, the merging of two relations (or classes) can be accommodated as long as data stored *subsequently* in the merged relation can be retrieved through the prior versions. For this to happen, extra fields may need to be included. However, in other cases the *vacuuming*[4] of versions may be required. Readers are referred to [24, 26] for more information.

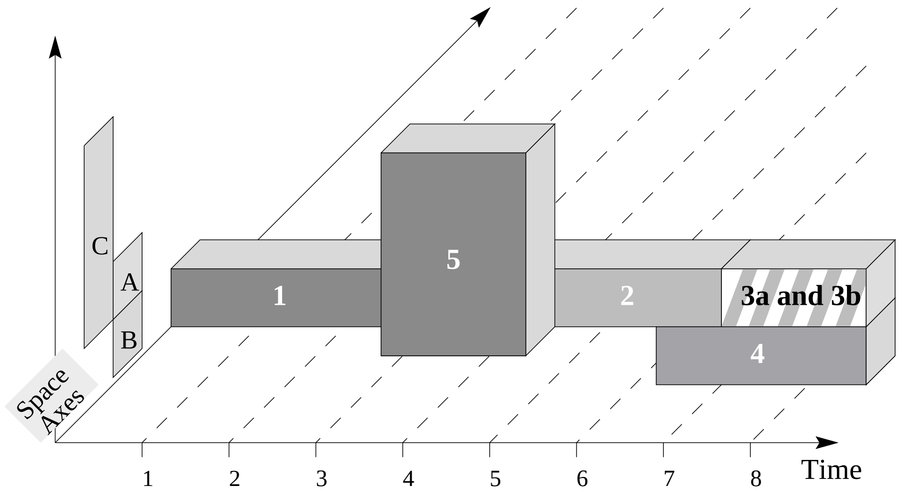

Figure 3. Representation of schema version pertinence considering space, time and label coordinates.

[4] Vacuuming of data and schema versions is used to permanently delete data or versions – something not normally permitted in temporal databases.

For example, in Figure 3, six schemas are defined, (we assume just one dimension for time for the purposes of the example but this can be expanded to further dimensions). Schema 1 was the first to be active and applied to transactions applicable to Area A. Schema 2 took over at some point and, following that, schemata 3A and 3B which are both valid for the same spatio-temporal region but which differ in some respect (one maybe, for example, a test schema). Schemata 4 and 5 are valid for different regions (B and C resp. as shown in the projection) and different times.

In our model, query specifications include spatio-temporal areas for data access (input schemas) as well as a target spatio-temporal area for final data formatting (output schema). Schemata are constructed for queries as follows:

1. The spatio-temporal area over which the query ranges is determined.

2. This area is unioned with the spatio-temporal area defined by the output schema (if not included in the area determined in step 1.).

3. The completed schema is constructed for that area. This provides a schema through which all data, including any query criteria, can be viewed losslessly.

4. The query is executed using the compiled schema coercing data to conform to the completed schema. (Note that no information that would not have otherwise have been lost anyway in the next step will be lost here).

5. The results are then converted according to the output transformation rules[5] to the output schema format as required.

Note that a bounding region for queries can be defined as the union of all the spatio-temporal areas involved. For instance, a query that takes in spatial regions A and B and multiple schemata temporally, say, times 5 through 7, would use for compilation of the answer, the completed schema for schemata, 2, 3A, 3B and 4. The problem thus

[5] TSQL2 discussed the use of special values when domains were restricted. For example, changes may be specified as follows:

```
ALTER TABLE Employee (
  Salary DECIMAL(5,0) INAPPLICABLE(99999) )

ALTER TABLE Employee (
  Name   CHAR(20) INAPPLICABLE(SUBSTRING(Name FROM 1 FOR 17 || "...")) )
```

becomes a general problem of schema integration coupled with the accommodation of temporal and spatial semantics.

3.2. Spatio-Temporal Schema Versioning with Parallel Versioning - A Model for Schema and Data Selection

The model we propose here is generic enough to be applied to relational, object-oriented databases, and others. Furthermore, it is a flexible model supporting seamless integration of temporal, spatial and parallel versioning facilities. Indeed, all the "versioning dimensions" are treated in a uniform manner, through a multi-dimensional coordinate system. At the same time, by means of "defaults" on some coordinates that imply special selection procedures, it can be used for temporal or spatial or parallel versioning only, or combinations (e.g. spatio-temporal only).

The model used for the purposes of this paper has been developed from, and is an extension of, the *Bitemporal Conceptual Data Model (BCDM)* presented by Jensen, Snodgrass and Soo in [15]. The BCDM has as its lowest element a bi-temporal chronon, which is then built into larger structures as needed. For each attribute, each collection of cells is then assigned a value from the domain or is null. Thus, for each tuple, the representation can be considered to be a sparse map of the values assigned in 2-dimensional time.

Our extension adds $2n$-dimensions[6] to the BCDM so that, for example, a model requiring bi-temporal support with bi-spatial 3D representation would be defined as an 8 dimensional hypercell. While this may be difficult to conceive, the elementary operations possible over the BCDM (insertion, deletion, etc.) have their analogues in this extended model.

On top of this base model we can name regions of space-time through simple enumeration. If multiple, parallel labelling is required (as would be required for overlapping spatio-temporal areas or the use of dual *production* and *development* schemata for one time and location) this can be considered as having the effect of adding one further dimension to the base model above. Thus, the conceptual data model we consider here is 9-dimensional, with a symbolic label for parallel versioning being the additional dimension. For the purposes of this paper we will adopt a 9-dimensional coordinate set, which can be defined as:

$$D_{9D} = D_L \times D_{TSx} \times D_{TSy} \times D_{TSz} \times D_{VSx} \times D_{VSy} \times D_{VSz} \times D_{TT} \times D_{VT}$$

[6] For the purposes of our proposals we require a model that accommodates multi-spatial as well as multi-temporal semantics.

Table II. Elementary coordinate domains

Domain	Meaning
$D_L = \{l_1, l_2, \ldots, l_i\}$	the domain of labels
$D_{VT} = \{t_1, t_2, \ldots, t_j\}$	the domain of valid times
$D_{TT} = \{t'_1, t'_2, \ldots, t'_{j'}\}$	the domain of transaction times
$D_{VSx} = \{x_1, x_2, \ldots, x_l\}$	the domain of valid space x coordinates
$D_{VSy} = \{y_1, y_2, \ldots, y_m\}$	the domain of valid space y coordinates
$D_{VSz} = \{z_1, z_2, \ldots, z_n\}$	the domain of valid space z coordinates
$D_{TSx} = \{x'_1, x'_2, \ldots, x'_{l'}\}$	the domain of transaction space x coordinates
$D_{TSy} = \{y'_1, y'_2, \ldots, y'_{m'}\}$	the domain of transaction space y coordinates
$D_{TSz} = \{z'_1, z'_2, \ldots, z'_{n'}\}$	the domain of transaction space z coordinates

where the single coordinate domains and their meanings can be found in Table II. In order to simplify the notation a little, we will use a more compact form:

$$D_{9D} = D_L \times \overline{D}_{BS} \times D_{BT}$$

where $D_{BT} = D_{TT} \times D_{VT}$ is a bitemporal domain and $\overline{D}_{BS}$ is a bispatial (tridimensional) domain.

For the purposes of the paper we will term the new model the Multi-Spatio- Temporal Data Model or MSTDM, however, it should be noted that its is not the aim of this paper to develop a powerful model or discuss the power of the model in depth. This is also supported by the fact that, when dealing with heterogeneous data sources, we cannot specialise the model to a great extent without losing in general applicability.

In our schema selection model, the database is composed of an intentional part (meta-data, that is a set of schema versions $SV_1, \ldots, SV_N$) and an extensional part (object data, that is a set of data repositories). Each schema version can be structured as a schema definition part plus an MSTDM "coordinate-stamp". The schema definition part includes all the data structure definitions (e.g. a set of catalogue tables in a relational database, a set of class definitions in an object-oriented database), while the coordinate part defines the multi-dimensional area representing the pertinence of the schema version. In our model, it is made of three sets of labels, spatial regions and temporal elements, respectively. Hence, each schema version SV_k can be defined as follows:

$$SV_k = (DefSV_k | P_k)$$

where $P_k \subseteq D_{9D}$. The coordinate system is then used for the schema version selection by means of the selection function SV. When the three coordinates are all supplied to represent a point in the versioning space, a single schema version is always selected (if it exists) as follows:

$$SV(l, \overline{s}_b, t_b) = \{SV_k | (l, \overline{s}_b, t_b) \in P_k\}$$

In general, more than one schema version can be used at the same time. This happens, for instance, when some of the coordinates are omitted or when a set of points rather than a single point is specified. In this case, a schema-version composition function ($\oplus$) is needed to obtain a completed schema. The composition function will typically be based on union or intersection semantics (i.e. $\oplus$ becomes $\cup$ or $\cap$, respectively), depending on application requirements. For instance, when the label coordinate is omitted, the selection function can be defined as:

$$SV(*, \overline{s}_b, t_b) = \bigoplus_{l \in D_L} SV(l, \overline{s}_b, t_b)$$

Notice that the result can be used as the complete schema version set of a database supporting only spatio-temporal schema versioning. In fact, a two- variable schema selection function can be defined as $SV'(\overline{s}_b, t_b) = SV(*, \overline{s}_b, t_b)$. Other coordinate combinations and single coordinate versionings can be dealt with in the same way.

On the other hand, intervals or set of coordinates can also be used for schema construction. The most general definition of the SV function is, thus:

$$SV(L, \overline{S}, T) = \bigoplus_{l \in L, \overline{s}_b \in \overline{S}, t_b \in T} SV(l, \overline{s}_b, t_b)$$

In fact, the previous examples can also be brought back to this definition.

However, the complete specification of the composition function $\oplus$ is, in general, more complicated than that of a simple union or intersection operation. It requires the application of complex schema and/or data integration techniques, which will be briefly discussed in Section 4.

As fas as the extensional part of the database is concerned, we address query processing assuming an integrated vision of the data is adopted. Such a vision could be based on either virtual or materialized views on actual data. We can consider (at logical level) data repositories organized as a common store augmented with at most N differential stores, if N is the number of schema versions of interest (see Figure 4). The common store has a logical structure corresponding to the definitions common to all the schema versions. For instance, if two schema versions contain two attribute definitions with the same name and storage-compatible types (e.g. integer and real numbers), just one

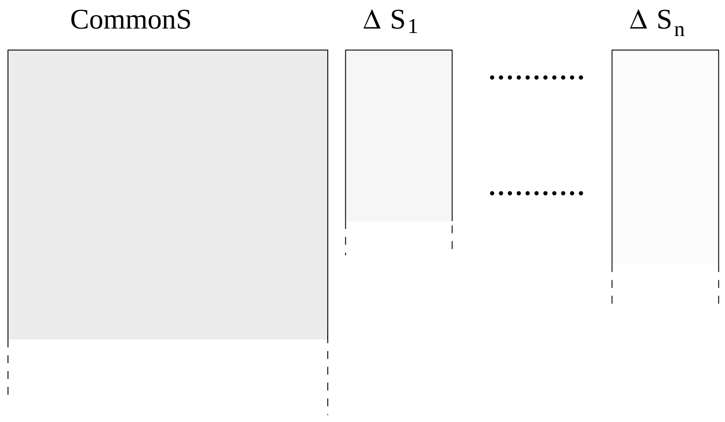

Figure 4. Logical organization of data with its corresponding structure.

attribute with the same name and the most “generic” type is included in the common store structure. Therefore, this structure can be defined as:

$$CommonS = \{X | \forall k, X \text{compatible with an element in } DefSV_k\}$$

The purpose of maintaining the common store is to improve, as much as possible, an integrated representation of data, to avoid redundancy and, if stores are materialized, replication. The differential stores are then organized with structure:

$$\Delta S_k = DefSV_k \setminus CommonS$$

The differential store is actually defined only if the above definition is not empty, that is when $DefSV_k$ does not contain only common parts. With this definitions, the data which are instances of the SV_k schema version can be retrieved from the common store and, if ΔS_k is not empty, from the corresponding differential store. Notice that the data retrieved from the common store may need a type conversion (e.g. an attribute in SV_k is an integer number which has been stored as a real number in the common store). A reference to a suitable conversion function can be stored in the differential store in this case.

Such an organization allows an efficient data retrieval when completed schemas are involved. Let us assume that data belonging to schema versions SV_1 and SV_2 are required. In this case, if the completed schema of SV_1 and SV_2 is computed with the intersection semantics, then the corresponding data can be found in the common store only, since the data stored in the differential stores that do not belong to both schema versions are discarded. On the other hand, if the completed schema is computed with the union semantics, the corresponding data can be found in the common store and in differential stores ΔS_1 and ΔS_2.

In general, the steps to be followed in query processing are those listed in Section 3. Once the label-spatio-temporal area (L, S, T) for the query has been determined, the corresponding completed schema QS can be constructed by means of the SV function as $QS = SV(L, S, T)$. Given this, data items can be accessed for the query from the common store and from the differential stores (corresponding to the schema versions involved in the QS construction). Such data are retrieved and processed according to the completed schemata, possibly after conversion functions from the differential stores have been applied.

4. Architectural Issues

In order to effect schema versioning in any practical manner, it is necessary to provide a mechanism for integrating different schemata. This corresponds to implementation of the $\oplus$ operator of our approach. Furthermore, since spatial versions may correspond to data stored on different geographic sites, the global scenario we consider is one of a multiplicity of heterogeneous sources. The goal of system interoperability and information integration is to provide a coherent view of data stored in multiple sources. This is necessary, in our case, to support multi-schema queries, possibly involving different sites. From the literature, two main approaches to integration problems are available [6]:

Procedural/Structural This approach is based on the provision of *mediators* and *wrappers*, and has been used, for instance, in `TSIMMIS` [9, 13]. In this architecture, wrappers that encapsulate data sources are employed to convert data objects to a common information model, whereupon mediators (which are software modules) combine the information provided by the wrappers to answer specific user-defined queries. To this end, they embody all the knowledge necessary for processing the required information. In general, as many mediators have to be coded as queries to answer.

Declarative/Semantic This approach is based on the definition of a *global schema* to obtain a unified representation of all the data. This is the approach taken in [2, 5, 6]. Each data source constructs its own conceptual schema, encoding semantic information, available to the outer world. Such schemata are converted to a common model and reconciled, so that multi-source queries can be issued at this level and automatically decomposed into single-source subqueries which are then executed at the local sites. A suitable mechanism (for example, one based on rewriting rules) is then used to put the returned information together according to the global schema. Wrappers are still present to convert data from the sources, whereas a single mediator may be present to manage the global schema (and interschema relationships) and perform the necessary query rewriting.

For our purposes, the choice of one of the two approaches is not central to the aims of this paper and will not be discussed in depth. However, both are feasible. The former solution could be used by defining a mediator for each query class of interest, by implementing the specialized $\oplus$ operator for the schema versions (and data sources) involved. A coordinator mediator, having an overall knowledge about the relevance of a given schema version, could redirect a query to the appropriate mediator. This approach would avoid the resolution of integration problems for data sources which would never occur together in useful queries. The latter solution could be adopted to easily implement query processing. Indeed, if we define the global schema as $SV(*, *, *)$ in our system, then any completed schema defined by a query can be considered a simple view on the global schema. Hence, one mediator is required to manage the global schema and the query rewriting process.

As far as extensional data are concerned, the distinction between the single common and multiple differential stores discussed in the previous section, can be applied here to the data processed by a mediator. With the procedural/structural arrangement, such data are those concerned by a query, whereas, with the declarative/semantic operation, they may represent all the information available. In the former case, the structure of the common and differential stores corresponds to the completed schema of the query, in the latter corresponds to the global schema. In both cases, all the stores contain directly interoperable data – that is data coming from the sources which have been suitably translated by wrappers into a common format and for which the mediator has resolved any conflicts.

5. Query Language Implications

In [21, 26] query language enhancements were suggested to accommodate temporal schema versioning. Apart from the more obvious extensions to cater for spatial queries themselves (which are covered elsewhere), we briefly discuss below extensions to support the proposals in this paper based on the enhancements outlined in the TSQL2 proposal [26].

5.1. Schema Specification

In TSQL2, schemata are specified through the use of an environmental, single point in time `SET SCHEMA` command. The minimal extension required would be to extend the `SET SCHEMA` command to include the spatial context. However, the use of a more complex `SET SCHEMA` command and of a `SCHEMA-TIME` clause within the main `SELECT` statement (as discussed in [26]) holds many benefits, including the ability to construct schemata within the query statement itself. For example, schemata could be constructed to cover two or more geographical regions or an interval in time. Provision for a separate output schema specification should not be forgotten too.

5.2. Version Naming

The concept of naming versions for ease of specification was discussed but omitted from the TSQL2 proposals. However, with the complex specification of spatial regions (a country for example), the textual specification becomes more important. For example, the specification of `SET SCHEMA AUSTRALIA` is far easier than the specification of the geographical coordinates.

6. Further Research

This paper discusses the first proposal to date to accommodate schema versioning in both multi-temporal and spatial databases. Research is continuing to refine both the model as well as associated issues such as query language support and concerns relating to architectural issues (such as lazy and strict data conversion) and speed of access, particularly for access to data through the *current* schemata. Further work will be devoted to the introduction, in our model, of spatio-temporal versioning also at the object instance level. Semantic issues and practical implications concerning the management of versioning both at the schema and data levels deserve a deeper investigation (e.g. the problem has been studied, for the temporal case, in [8]).

Our consultancy experience together with discussions with other researchers has indicated, albeit anecdotally, that spatial schema versioning would be a useful adjunct to many systems. It would interesting to investigate this utility further to discover the extent and nature to which spatial schema versioning could be useful.

7. Acknowledgements

This work was conceived while the authors attended the Workshop on Integrating Spatial and Temporal Databases at Schloss Dagstuhl in November 1998. We would like to thank IBFI GmbH for their kind hospitality and for providing a conducive atmosphere in which to work. This work also represents an extension on that presented at the Workshop on Spatio-Temporal Data Models and Languages in Florence [25]. We are similarly grateful for the useful comments we received there, and for the additional anonymous suggestions of the *Geoinformatica* referees which helped to improve this paper.

References

1. Abraham, T. and J. Roddick: 1999, 'Survey of spatio-temporal databases'. *Geoinformatica* **3**(1), 61–99.
2. Arens, Y., C. Knoblock, and W. Chen: 1996, 'Query Reformulation for Dynamic Information Integration'. *Journal of Intelligent Information Systems* **6**(2), 99–130.
3. Ariav, G.: 1991, 'Temporally oriented data definitions: managing schema evolution in temporally oriented databases'. *Data and Knowledge Engineering* **6**(6), 451–467.
4. Banerjee, J., H.-T. Chou, H. Kim, and H. Korth: 1986, 'Schema evolution in object-oriented persistent databases'. In: *Sixth Advanced Database Symposium.* Tokyo, pp. 23–31.
5. Bergamaschi, S., S. Castano, and M. Vincini: 1999, 'Semantic Integration of Semistructured and Structured Data Sources'. *SIGMOD Record* **28**(1), 54–59.
6. Cavalnese, D., G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati: 1998, 'Information Integration: Conceptual Modeling and Reasoning Support'. In: *Third IFCIS International Conference on Cooperative Information Systems (CoopIS).* New York City, NY, pp. 280–291.
7. Clifford, J. and D. Warren: 1983, 'Formal semantics for time in databases'. *ACM Transactions on Database Systems* **8**(2), 214–254.
8. De Castro, C., F. Grandi, and M. Scalas: 1997, 'Schema Versioning for Multitemporal Relational Databases'. *Information Systems* **22**(5), 249–290.
9. Garcia-Molina, H., Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom: 1997, 'The TSIMMIS approach to mediation: data models and languages'. *Journal of Intelligent Information Systems* **8**(2), 117–132.
10. Grandi, F., F. Mandreoli, and M. Scalas: 2000, 'A Generalized Modeling Framework for Schema Versioning Support'. In: *Eleventh Australasian Database Conference.* Canberra, Australia, pp. 33–40.
11. Günther, O. and A. Buchmann: 1990, 'Research Issues in Spatial Databases'. *SIGMOD Record* **19**(4), 61–68.
12. Güting, R. H.: 1994, 'An Introduction to Spatial Database Systems'. *VLDB Journal* **3**(4), 357–399.

13. Hammer, J., H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig, and V. Vassalos: 1997, 'Template-based wrappers in the TSIMMIS system'. *SIGMOD Record* **26**(2), 532–535.
14. Jensen, C., C. Dyerson, M. Bhlen, J. Clifford, R. Elmasri, S. Gadia, F. Grandi, P. Hayes, S. Jajodia, W. Kfer, N. Kline, N. Lorentzos, Y. Mitsopoulos, A. Montanari, D. Nonen, E. Peressi, B. Pernici, J. Roddick, N. Sarda, M. Scalas, A. Segev, R. Snodgrass, M. Soo, A. Tansel, P. Tiberio, and G. Wiederhold: 1998, 'A consensus glossary of temporal database concepts - February 1998 Version'. In: O. Etzion, S. Jajodia, and S. Sripada (eds.): *Temporal Databases - Research and Practice*, Vol. 1399 of *Lecture Notes in Computer Science*. Berlin Heidelberg: Springer-Verlag, pp. 367–405.
15. Jensen, C., R. Snodgrass, and M. Soo: 1995, 'The TSQL2 data model'. In: R. Snodgrass (ed.): *The TSQL2 Temporal Query Language*. Norwell, Mass: Kluwer Academic Publishers, pp. 157–240.
16. Katz, R.: 1990, 'Toward a unified framework for version modeling in engineering databases'. *ACM Computing Surveys* **22**(4), 375–408.
17. Paredaens, J.: 1995, 'Spatial Databases, The Final Frontier'. In: G. Gottlob and M. Y. Vardi (eds.): *Database Theory*, Vol. 893 of *Lecture Notes in Computer Science*. Springer-Verlag.
18. Penney, D. and J. Stein: 1987, 'Class modification in the GemStone object-oriented DBMS'. *OOPSLA '87 (SIGPLAN Notices)* **22**(12), 111–117.
19. Phillips, A., I. Williamson, and I. . Ezigbalike: 1998, 'The Importance of Metadata Engines in Spatial Data Infrastructures'. In: *AURISA '98*. Perth, Western Australia, pp. 23–27.
20. Roddick, J.: 1991, 'Dynamically changing schemas within database models'. *Australian Computer Journal* **23**(3), 105–109.
21. Roddick, J.: 1992, 'SQL/SE - a query language extension for databases supporting schema evolution'. *SIGMOD Record* **21**(3), 10–16.
22. Roddick, J.: 1995, 'A survey of schema versioning issues for database systems'. *Information and Software Technology* **37**(7), 383–393.
23. Roddick, J.: 1996, 'A model for schema versioning in temporal database systems'. *Australian Computer Science Communications* **18**(1), 446–452.
24. Roddick, J., N. Craske, and T. Richards: 1993, 'A taxonomy for schema versioning based on the relational and entity relationship models'. In: R. Elmasri, V. Kouramajian, and B. Thalheim (eds.): *Twelfth International Conference on Entity-Relationship Approach*. Dallas, Texas, pp. 143–154.
25. Roddick, J., F. Grandi, F. Mandreoli, and M. Scalas: 1999, 'Towards a model for spatio-temporal schema selection'. In: *Workshop on Spatio-Temporal Data Models and Languages (W06 - Tenth International Workshop on Database and Expert Systems Applications)*. Florence, Italy, pp. 434–440.
26. Roddick, J. and R. Snodgrass: 1995, 'Schema versioning support'. In: R. Snodgrass (ed.): *The TSQL2 Temporal Query Language*. Boston: Kluwer Academic Publishing, pp. Ch. 22. 427–449.
27. Snodgrass, R. (ed.): 1995, *The TSQL2 Temporal Query Language*. New York: Kluwer Academic Publishing.
28. Tan, L. and T. Katayama: 1989, 'Meta operations for type management in object-oriented databases - a lazy mechanism for schema evolution'. In: W. Kim, J.-M. Nicolas, and S. Nishio (eds.): *First International Conference on Deductive and Object-Oriented Databases, DOOD '89*. Kyoto, Japan, pp. 241–258.

Authors' Vitae

John F. Roddick
John Roddick currently holds the SACITT Chair of Information Technology in the School of Informatics and Engineering at the Flinders University of South Australia. He has also held positions at the Universities of South Australia and Tasmania, and as a project leader and consultant in the Information Technology industry. His technical interests include data mining and knowledge discovery, schema versioning and enterprise systems. He holds a PhD from La Trobe University, an MSc from Deakin University and a BSc(Eng)(Hons) from Imperial College, London. He is editor-in-chief of the *Journal of Research and Practice in Information Technology*, a fellow of the Australian Computer Society and the Institution of Engineers, Australia and a member of the IEEE Computer Society and the Association for Computing Machinery.

Fabio Grandi
Fabio Grandi is currently an Associate Professor in the Faculty of Engineering of the University of Bologna, Italy. Since 1989 he has worked at the CSITE (formerly CIOC) center of the Italian National Research Council (CNR) in Bologna in the field of neural networks and temporal databases. For this work he was initially supported by a fellowship from the CNR. In 1993 and 1994 he was an Adjunct Professor at the Universities of Ferrara, Italy, and Bologna. He joined his current department (Dept. of Electronics, Computer Science and Systems) as a Research Associate in 1994. His scientific interests include temporal databases, storage structures, access cost models, world wide web extensions. He received a Laurea degree in Electronics Engineering and a PhD in Electronics Engineering and Computer Science from the University of Bologna.

Federica Mandreoli
Federica Mandreoli is currently a PhD student at the Department of Electronics, Computer Science and Systems of the University of Bologna. She holds a Laurea degree in Computer Science from the same University. Her scientific interests are in the fields of spatio-temporal databases, object-oriented databases and schema versioning.

Maria Rita Scalas
Maria Rita Scalas is currently an Associate Professor in the Faculty of Engineering of the University of Bologna, Italy. From 1975 to 1979 she worked at the Universities of Pisa and Bologna supported by a

fellowship from the Italian Ministry of Education. In 1980 she became a Research Assistant in Computer Science at the University of Bologna and a consultant at the CIOC-CNR center of the National Research Council in Bologna. In 1986 she was a visiting scientist at the IBM Scientific Center in Heidelberg, Germany, where she took part in the AIM-P project. In 1987 she became an Associate Professor at the University of Trieste, Italy. She holds a Laurea degree in Physics from the University of Bologna. Her research interests are in the area of database management systems, temporal databases, access structures, optimizers and database design.