Towards a Model for Spatio-Temporal Schema Selection

John F. Roddick
School of Computer and Information Science, University of South Australia,
The Levels Campus, Mawson Lakes, South Australia 5095, Australia.
Email: roddick@cis.unisa.edu.au

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

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 is formatted also becomes an issue. This paper presents a generalised model that accommodates schema versioning within static, temporal, spatial and spatio-temporal relational and object-oriented databases.

Keywords: Schema Evolution, Schema Versioning, Spatio-Temporal Databases.

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 to handle change in the real world 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 or deletion of methods, etc. should be achievable with the minimum impact to both the data and to the applications. It also provides a mechanism for versions of a database structure to facilitate alternative structural representations to the same data. A previous survey of the field [22] provides a review of the area and a glossary of temporal database concepts [11] 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, 9, 10, 16] 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-related 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 that a given schema is to apply and the ability to apply different schemata according to where the data is to be applicable. For example, the differences applicable across land administration authorities (qv. [18]).

This paper therefore investigates the lessons learnt from the theory and development of temporally oriented schema versioning and examines whether these 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.

The paper is structured as follows. Section 2 reviews temporal schema versioning and provides some background to the model to be presented later. Section 3 then investigates the analogies between the temporal and spatial domains and provides some motivation for spatial schema versioning. Section 4 proposes a generalised spatio-temporal schema selection model and further discussion and research directions are provided in Section 5.

2. 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 concepts of a completed schema was developed following the ideas of Clifford and Warren [6] for a completed relation to enable all data, regardless of time of validity, to be accessed. The concept of a completed schema, which was introduced in [20] and discussed in more detail in [23], is an overarching schema through which all data, regardless of time of validity *or* format, to 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 is simply coerced to the new format as in [17]. Secondly, a lazy conversion mechanism can be used in which data is converted only when accessed [25]. Thirdly, data is never physically converted and is always accessed through conversion interfaces [4].

Finally, two query language extensions have been proposed which accommodate schema evolution, SQL/SE [20] and TSQL2 [21]. 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 is 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. Schema-time defaulted to transaction-time when not specified but could 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.

In [7], the single schema-time dimension (which was effectively a transaction-schema-time) of TSQL2 was expanded to support both transaction-schema-time and valid-schema-time. 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.

```
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

- **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 [21] 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 programs were compiled and thus the Transaction-schema-time applicable to any static data structures used by the program.
- **Decision-time.** The time the decision was taken to invoke the change, cause an event etc.

The most significant in terms of this paper are the first four which can be represented simply as shown in Figure 2.

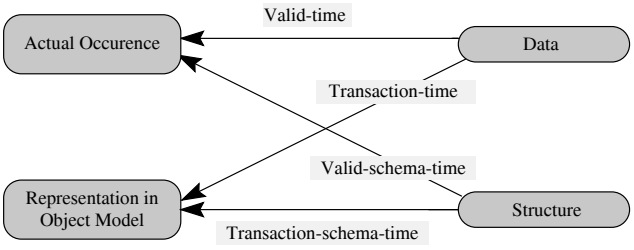

Figure 2. Data and Structure Time Dimensions

To augment the diagram given in [19], temporal schema dimensions can be defined (as well as the two temporal data dimensions) as shown in Figure 3.

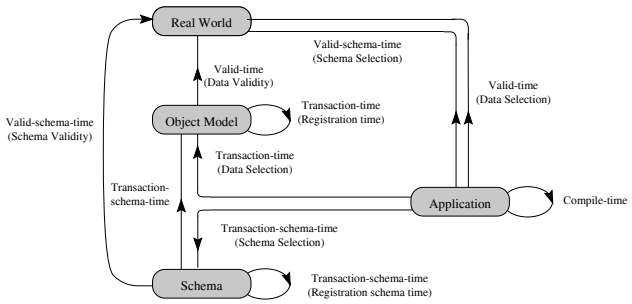

Figure 3. Real World, Object Model and Schema Time Dimensions

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 although in some 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 identical.

3. Spatial Schema 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.
- 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.

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 a popular area of research and the need for adequate schema versioning problem is becoming evident for these systems also. For example, spatial schema versioning would provide:

- a mechanism for different schemata to be applicable according to location. This may be important for jurisdictional, legal or collection method purposes;
- an indication of the rules under which data is collected and therefore the manner in which they should be interpreted. 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.

For example, consider a spatial system of rural maps in which 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,

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

[2] Ignoring quantum space-time.

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 needs 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 in Section 3 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.

4. A Model for Spatio-Temporal Schema Selection

4.1. 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 [21]). As we would want, the limit case is degradation to no versioning.

The model embraces the concept of a completed schema introduced 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.

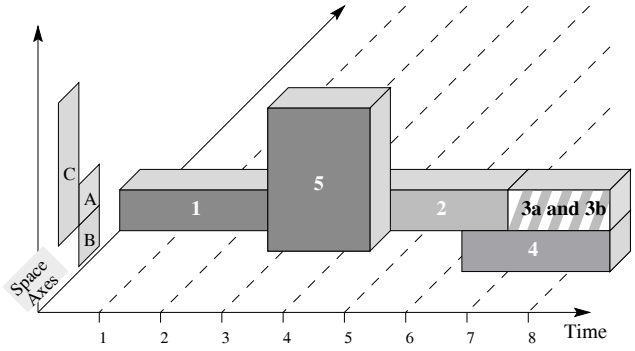

Figure 4. Representation of schema version pertinence considering space, time and label coordinates

For example, in Figure 4 six schemata are defined (we assume just one dimension for time for the purposes of the exemple 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 then 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.

Schemata are constructed for queries are follows:

1. The S/T area over which the query ranges is determined.
2. This area is unioned with the S/T 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 to the output schema format as required.

Note that queries also have a bounding region. 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 has thus becomes a general problem of schema integration coupled with the accommodation of temporal and spatial semantics.

4.2. 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).

In our 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 a "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. Notice that the temporal and spatial coordinates can also be themselves multidimensional (e.g. valid-location xyz coordinates and bitemporal time points). Hence, each schema version SV_i can be defined as follows:

$$\begin{aligned} SV_i &= (DefSV_i | L_i, S_i, T_i) \\ &= (DefSV_i | \{l_{i1}, \ldots, l_{in_i}\}, \{s_{i1}, \ldots, s_{im_i}\}, \\ &\quad \{t_{i1}, \ldots, t_{ip_i}\}) \end{aligned}$$

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:

$$SV(l, s, t) = \{SV_i | l \in L_i, s \in S_i, t \in T_i\}$$

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:

$$SV(*, s, t) = \bigoplus_{l \in \cup L_i} SV(l, s, t)$$

Notice that the result can be used as the complete schema version set of a database only supporting spatio-temporal schema versioning. In fact, its (two-variable) schema selection function can be defined as $SV'(s, t) = SV(*, s, t)$. 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:

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

In fact, also the previous examples can be reconducted 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 integration techniques. This issue has already been investigated in the literature. An overview of methods for schema integration in relational and semantic databases can be found in [5]. In the object-oriented field, although the problem is still an active research area, solutions have been proposed [26] where semantic and syntactic aspects are considered.

As fas as the extensional part of the database is concerned, different data organizations can be adopted. The organization we describe in the following (at logical level) is aimed to efficiently support query processing. We organize the data repository as a common store augmented with at most n differential stores, if n is the number of schema versions (see Figure 5). 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 attribute with the same name and the most "generic" type is included in the common store structure. Therefore, this structure can be defined as:

$$CommonS = \{X | \forall i, X \text{compatible with element in } DefSV_i\}$$

The purpose of maintaining the common store is to reduce, as much as possible, data replication. The differential stores are then organized with structure:

$$\Delta S_i = DefSV_i \setminus CommonS$$

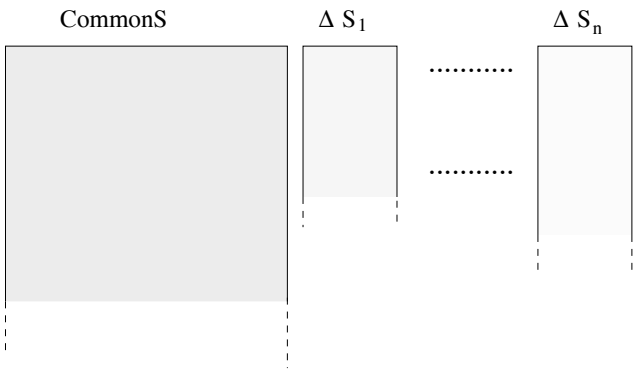

Figure 5. Organization of data with its corresponding structure

The differential store is actually defined only if the above definition is not empty, that is when $DefSV_i$ does not contain only common parts. With this definitions, the data which are instances of the SV_i schema version can be retrieved from the common store and, if ΔS_i is not empty, from the the corresponding differential store. Notice that the data retrieved from the common store may need a type conversion (e.g. an attribute in SV_i 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 schemata 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 DS_1 and DS_2.

In general, the steps to be followed in query processing are those listed in Section 4. Once the L/S/T area for the query has been individuated, the corresponding completed schema QS can be constructed by means of the SV function as $QS = SV(L, S, T)$. Then data are accessed for the query from the common store and from all 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.

5. Further Research

This paper discusses some initial ideas in what we believe is the first attempt 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.

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.

6. Acknowledgements

This work was conceived while the authors attended the Integrating Spatial and Temporal Databases Workshop 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.

We would also like to thank the referees for their helpful comments.

References

[1] Abraham, T. and Roddick, J.F. Survey of spatio-temporal databases. *Geoinformatica*, 3(1):In press. 1999.

[2] Andany, J., Leonard, M. and Palisser, C. Management of schema evolution in databases. In Proc. *Seventeenth International Conference on Very Large Databases*, 161-170. 1991.

[3] Ariav, G. Temporally oriented data definitions: managing schema evolution in temporally oriented databases. *Data Knowl. Eng.*, 6(6):451-467. 1991.

[4] Banerjee, J., Chou, H.-T., Kim, H.J. and Korth, H.F. Schema evolution in object-oriented persistent databases. In Proc. *Sixth Advanced Database Symposium*, 23-31. 1986.

[5] Batini, C., Lenzerini, M. and Navathe, S. B. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323-364. 1986.

[6] Clifford, J. and Warren, D.S. Formal semantics for time in databases. *ACM Trans. Database Syst.*, 8(2):214-254. 1983.

[7] De Castro, C., Grandi, F. and Scalas, M.R. Schema Versioning for Multitemporal Relational Databases. *Inf. Syst.*, 22(5):249-290. 1997.

[8] Grandi, F., Mandreoli, F. and Scalas, M.R. A Formal Model for Temporal Schema Versioning in Object-Oriented Databases. Technical Report CSITE-014-98, CSITE-CNR, Università di Bologna, 1998.

[9] Günther, O. and Buchmann, A. Research Issues in Spatial Databases. *SIGMOD Rec.*, 19(4):61-68. 1990.

[10] Güting, R.H. An Introduction to Spatial Database Systems. *VLDB Journal*, 3(4):357-399. 1994.

[11] Jensen, C., et al. A consensus glossary of temporal database concepts - February 1998 Version. In *Temporal Databases - Research and Practice*, Lecturer Notes in Computer Science, 1399, Springer-Verlag, Berlin Heidelberg. 1998.

[12] Kim, H.-J. and Korth, H.F. Schema versions and views in object-oriented databases. In Proc. *InfoJapan '90: Information Technology Harmonizing with Society*, 277-284. 1990.

[13] Monk, S.R. and Sommerville, I. Schema evolution in OODBs using class versioning. *SIGMOD Rec.*, 22(3):16-22. 1993.

[14] Nguyen, G.T. and Rieu, D. Schema evolution in object-oriented database systems. *Data Knowl. Eng.*, 4(1):43-67. 1989.

[15] Orlowska, M.E. and Ewald, C.A. Schema evolution - the design and integration of fact-based schemata. In *Research and Practical Issues in Databases, Third Australian Database Conference* World Scientific, La Trobe University. 1992.

[16] Paredaens, J. Spatial Databases, The Final Frontier. In *Database Theory* Springer-Verlag, 1995.

[17] Penney, D.J. and Stein, J. Class modification in the GemStone object-oriented DBMS. *OOPSLA '87 (SIGPLAN Notices)*, 22(12):111-117. 1987.

[18] Phillips, A., Williamson, I.P. and Ezigbalike, I.C. The Importance of Metadata Engines in Spatial Data Infrastructures. In Proc. *AURISA '98*, 23-27. Perth, Western Australia. 1998.

[19] Roddick, J.F. Dynamically changing schemas within database models. *Aust. Comput. J.*, 23(3):105-109. 1991.

[20] Roddick, J.F. SQL/SE - a query language extension for databases supporting schema evolution. *SIGMOD Rec.*, 21(3):10-16. 1992.

[21] Roddick, J.F. and Snodgrass, R.T. Schema versioning support. In *The TSQL2 Temporal Query Language* Kluwer Academic Publishing, Boston. 1995.

[22] Roddick, J.F. A survey of schema versioning issues for database systems. *Inf. Softw. Technol.*, 37(7):383-393. 1995.

[23] Roddick, J.F. A model for schema versioning in temporal database systems. *Aust. Comput. Sc. Commun.*, 18(1):446-452. 1996.

[24] Snodgrass, R.T. (ed.) *The TSQL2 Temporal Query Language*. Kluwer Academic Publishing, New York. 1995.

[25] Tan, L. and Katayama, T. Meta operations for type management in object-oriented databases - a lazy mechanism for schema evolution. In Proc. *First International Conference on Deductive and Object-Oriented Databases, DOOD '89.*, 241-258. 1989.

[26] Thieme, C. and Siebes, A. Schema integration in object-oriented databases. In *Conference on Advanced Information System Engineering (CAiSE'93)*, Paris, France, 54-70. 1993.