Spatio-Temporal Databases: Contentions, Components and Consolidation

Norman W. Paton, Alvaro A.A. Fernandes and Tony Griffiths
Department of Computer Science
University of Manchester
Oxford Road, Manchester M13 9PL, UK
(norm, alvaro, griffitt)@cs.man.ac.uk
http://img.cs.man.ac.uk

Abstract

Spatio-temporal databases have been the focus of considerable research activity over a significant period. However, there are as of yet very few prototypes of complete systems, far less products that provide effective support for applications tracking changes to spatial and aspatial data over time. We contend that this is because much of the activity in spatio-temporal databases has focused on specific parts of the problem, at the expense of a more holistic view of database systems design and development. It is probably also the case that the database research community has been inclined to undervalue integration or consolidation activities. This paper outlines some contentions relating to spatio-temporal databases, with a view to pruning the space of possible paths that consolidation activities might follow. Suggestions are also made as to what areas are most likely to present challenges to a consolidation activity, in the light of a model architecture for a spatio-temporal database.

1. Contentions

This section presents a number of contentions relating to research in spatio-temporal databases. By definition, not everyone will agree with all of these, but it is important that a research community periodically pauses to reflect on where it is going and why.

1. *Spatio-Temporal databases are not (just) about moving objects.* There has been significant recent activity on moving objects (e.g., [3]); such work seems likely to yield results that are of value to real applications. However, it is by no means obvious that moving object proposals will address all the requirements of applications modelling changes to objects that are not directly associated with movement (e.g., boundaries, rivers).

2. *Spatial and aspatial data change in similar ways.* This contention essentially proposes that a single temporal model should be usable with both spatial and aspatial data.

3. *There is more to query processing than indexes.* There has been much more research carried out on spatio-temporal indexes than on other features of query processors – query algebras, optimisers, join algorithms, etc, even though these components are crucial to the development of complete spatio-temporal database systems.

4. *Rampant featurism has been a significant impediment to real progress.* Proposals for spatial and temporal database models and languages often have very large numbers of features, which together would be extremely difficult to implement. More is not always better, and proposals that are difficult to implement may impede the development of leaner, more practical approaches.

5. *You can't please all of the people any of the time ... so it is enough to start by pleasing someone.* This follows on from the previous contention. In the absence of practical spatio-temporal database systems, users of spatial and temporal data are making do with much more generic facilities. A significant number of tasks can probably be given effective support by reasonably lean, but nevertheless fully developed, spatio-temporal systems.

6. *It is easier to propose a spatio-temporal model than to implement it.* We assume that this is the reason why there are so many more proposals than prototypes. Researchers should be slow to propose models that they are not prepared to prototype (and thereby to evaluate in practice).

7. *Spatio-temporal databases are not descended from geographical information systems.* The right place to start building a spatio-temporal database system is an aspatial database system, not a GIS. Adding facilities that are characteristic of temporal databases (e.g., declarative queries) to a GIS is likely to be a rocky road to a cumbersome proposal.

The above contentions motivate a view that research in spatio-temporal databases could benefit from a period of consolidation, in which the focus is on the development of comprehensive working prototypes. Such an emphasis can be used to identify areas that present open research challenges that must be addressed before effective spatio-temporal database systems can be developed. Such an emphasis also focuses the mind in terms of the most important kinds of functionality – it is unlikely that scalable working prototypes can be developed in the short to medium term that support all of vector and raster data, valid and transaction time, spatial and temporal indeterminacy, moving objects and constraint programming. Thus identifying functionalities that together are useful for significant categories of application, and focusing on them, should allow individual research activities to identify important challenges that are grounded on well defined application needs.

The remainder of the paper is structured as follows. Section 2 presents a database architecture, and discusses where extensions are required to that architecture for supporting spatial and temporal data. Section 3 gives some suggestions as to the principal issues that remain open in the light of the architecture in Section 2.

2. Components

This section considers how a spatio-temporal DBMS might be designed and implemented using a component-based approach and an architectural framework that reflects current, mainstream database technology.

An Architectural View of DBMSs We propose an architectural framework (depicted in Figure 1) in which DBMSs are viewed as comprising programming language interfaces, a query processing component, and a services manager (which, we assume, is responsible for the management of storage structures, security, concurrency, transactions, recovery, etc.).

The programming language interfaces allow application programs to access and manipulate data using the abstractions of the data model implemented by the DBMS.

The query processor (depicted in more detail in Figure 2) is assumed to comprise a compiler (from a surface syntax to a logical algebra), a logical optimiser (which uses equivalences to rewrite logical algebraic expressions into forms

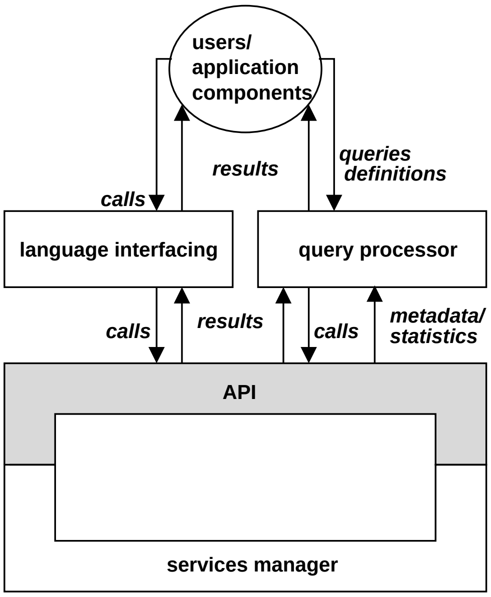

Figure 1. A Component-Based DBMS Architecture

that are, heuristically, more efficient to evaluate), a physical optimiser (which, in the light of the available access methods and indices, uses statistics about the data to convert logical algebraic operators into physical algebraic ones, thereby generating a query plan), and an evaluator (which traverses the query plan making calls to algorithms and access methods to compute the answer).

The services manager is assumed to be visible via an application program interface (API) which serves data and metadata to application programs and to the query processor whilst enforcing, e.g., concurrency, transaction, recovery and security controls.

Spatial and Temporal Extensions We now consider the problem of specifying, designing and implementing spatial and temporal DBMS by extending an architecture such as that depicted in Figures 1 and 2. The targeted DBMS is expected to support DBMS functionality over spatial and temporal data orthogonally and synergistically.

By orthogonality we mean that users can model the spatial features of types and properties, or the temporal aspects of modelled types and properties, or both, or neither.

By synergy we mean that if a user makes use of either only spatial or only temporal or both spatial and temporal facilities, the system responds positively (e.g., by making available specific syntax, by providing additional behaviour, by seeking optimisation opportunities, etc.).

The most essential decision in this approach to extensions is, of course, how the aspatial, snapshot data model

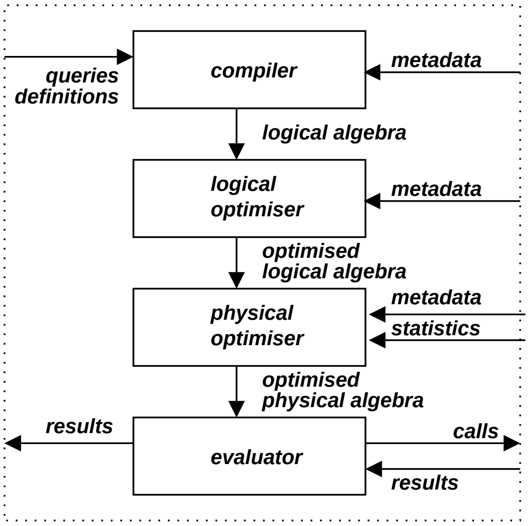

Figure 2. The Query Processor in Figure 1

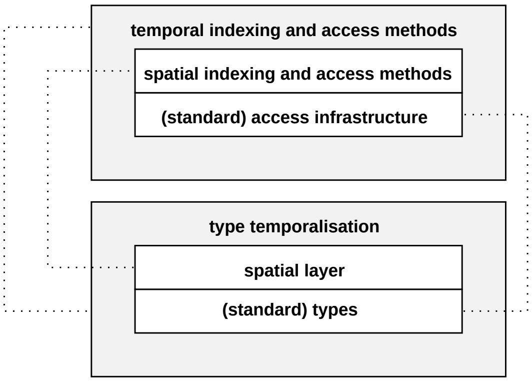

Figure 3. The Storage Manager in a Spatio-Temporal Services Manager Component

is to be impacted. We note that, at the level of the data model, extending an aspatial database (temporal or not) to support spatial applications only requires extending the set of supported types (to include, say, points, lines and regions). By contrast, extending a snapshot database (spatial or not) to support temporal applications requires not only extending the set of supported types (to include, say, instants and intervals), but also making provisions for all (or at least many) data model types to be associated with a history. In a sense, spatial extensions need induce no changes to the pre-extension types, whereas temporal extensions imply that the way the DBMS used to handle types is different from the way that it is expected to handle them now.

Figuratively speaking, spatial extensions can be layered over pre-extension types, but temporal extensions, at the very least, wrap around all supported types. The same applies, of course, to index and access method infrastructures. This view of the world is illustrated in Figure 3.

Implications of Spatial Extensions Extensions to support spatial applications are backward compatible with the set of types provided by the systems being extended in the sense that their specification, design and implementation depends on that of the existing types (i.e., integers, strings, etc.) but does not alter them at any level of abstraction (i.e., conceptually, logically or physically).

Given a set of spatial types, their formal characterisation (e.g., as carriers of a spatial algebra) defines well-formedness conditions for the insertion and deletion of values in database states, relationships between types (which can inform the design of spatial indexing mechanisms and optimisation strategies), and, most visibly, the operations that can be used to access and mutate values, as well as to test for conditions and to map values of one type into values of some other type.

This characterisation of spatial types as new primitive types gives rise on the one hand to a concrete syntax (for data definition and manipulation) and to a logical algebra. The spatial logical algebra is integrated with the existing logical algebra as yet another primitive algebra (on a par, therefore, with the integer algebra, the string algebra, etc,) among those that the services manager supports. Thus, just as addition on integers can be operator nodes in a query tree, so can, say, union of regions. Likewise, just as a less-than predicate on integer values can appear as a selection condition, so can, say, an inside predicate on, e.g., a lines and a regions value. A representative spatial algebra is described in [5].

For a services manager to support a spatial algebra it must implement storage structures for spatial values, algorithms (denoted by physical algebraic operators) for the spatial operations, and auxiliary structures (such as spatial indices). Therefore, new services are added or extended, but no existing services need to be significantly disrupted. As Figure 3 indicates, spatial extensions can, roughly speaking, be layered over existing functionality.

With respect to Figure 2, each and every module of the query processor is affected, but only by minimally disruptive extensions. Thus the compiler will recognise more literals, more function identifiers, and there will be more typing restrictions to verify and enforce. The logical and physical optimisers, as well as the evaluator, will range over an enlarged set of operators and will require new heuristics, and new cost modelling parameters in order to respond to the greater number of opportunities for rewriting and algorithm selection that arise.

Finally, with respect to Figure 1, the language interfaces will be enriched with new signatures corresponding closely to the spatial algebraic specification adopted.

Implications of Temporal Extensions Over and above analogous implications in each and all of the dimensions discussed above for the spatial case, temporal extensions also imply changes over the services previously provided for pre-extension types.

As in the spatial case, a set of temporal types (e.g., instants and intervals) must be formally characterised (e.g., as carriers of a temporal algebra) and, as described above, query processor components as well as language interfaces will be affected analogously.

Note that this only suffices for an application type (say, employee) to be associated with a property (say, employment-period, describing the (possibly closed) interval of time in which the employee has been with the employing organisation).

However, temporal extensions imply more functionality. In particular, the previous paragraph still characterises a snapshot database in that the only change to the snapshot model is an increase in the number of primitive types that are supported. In contrast to this, of course, a temporal DBMS keeps (if required) historical information, and thus information about the past is maintained by the system, and queries can take such information into account.

To support this richer view of how the objects modelled in the database are changing with time, temporal extensions impact all pre-extension types (including spatial types if they happen to be supported). For example, salary (of type, say, integer) may be temporalised (by which we mean that not just current but previous values too are maintained in a history). Similar considerations apply to values of any other type (e.g., address of type string, border of type region, etc.). The pervasiveness of temporalised types is illustrated in Figure 3 by temporal extensions being wrapped around all other supported types.

The type temporalisation referred to at the bottom of Figure 3 can conceptually be understood as the implicit availability of a new collection type for modelling histories, such that each element in a history (of an application type extent, property or relationship) pairs a value (of the extent, property or relationship) being modelled with a temporal value. In addition, histories come with an interface consisting of accessor, filtering and conversion (but not necessarily constructor, mutator or destroyer) operations. The operations on histories in turn give rise to extensions to the logical and physical algebras of the query processor illustrated in Figure 2. An example of a logical temporal algebra is given in [2].

Temporal extensions thus have significant consequences for the architecture of a database, in that the management of historical information on all types in the model implies significant revisions to the services manager and to the query processor.

Implications of Spatial plus Temporal Extensions Given the implications of spatial and of temporal extensions, which were considered in isolation above, what, if any, are the additional implications of having spatial types supported within a temporal DBMS? Strictly, the provision of a collection of spatial types within a temporal database should yield a coherent spatio-temporal database.

Assuming, for the meantime, that a spatio-temporal database needs no operations over and above those associated with the spatial and temporal types, there may still be additional implications for the DBMS if good performance is to be obtained. For example, if a spatio-temporal index is to be used (e.g., [6]) to improve the performance of queries involving space *and* time, then this implies an extension to (at least) the physical algebra and optimiser, as well as the obvious introduction of the index to the services manager.

Furthermore, it is likely that, with a view to minimising the use of disk space for temporally evolving spatial data, specialised storage structures can be expected to be of benefit. For example, a storage manager could choose to record the modifications that have been made to a spatial value, rather than copying a large spatial value every time a small change is made.

Returning to the question of language extensions, it is possible, perhaps likely, that spatio-temporal databases would benefit from the provision of some language constructs in addition to those provided with independently conceived spatial and temporal extensions. These are not likely to be numerous, but arguments have been advanced to the effect that additional language constructs are useful in certain cases (e.g., [4]). Such constructs clearly require support through surface syntax, algebras and evaluation algorithms.

3. Consolidation

This section briefly reviews the architecture of Section 2, discussing the areas that require significant attention from the research community if scalable working prototypes are to be produced. The aim of this section is not to propose specific models for use in a spatio-temporal database, as study of different categories of application are likely to yield different ways forward, but rather to argue that a consolidation based approach does yield well defined and important topics for research. The authors are currently involved in the development of a spatio-temporal database system, in which the aspatial data model is the ODMG object model [1], the spatial data model is that of the ROSE algebra [5], and the valid time data model is essentially that of the 2D ROSE algebra in 1D. The project is not currently addressing issues relating to moving objects or indeterminacy.

Languages Figure 1 identifies two language interfaces to a DBMS – a query language interface and a programming language interface, both of which share the type system provided by the data model of the DBMS. There has been much work on data models and user level languages for both spatial and temporal constructs. As we believe that there are relatively few additional constructs associated with spatio-temporal systems, it should be feasible to design coherent spatio-temporal data models and languages by building on earlier results. The open research challenges involve identifying new constructs that are of particular use where both spatial and temporal data are being accessed or manipulated.

Services There has been a significant amount of work on the development of spatio-temporal indexes, and while there is doubtless scope for further developments, this area can be considered to be more mature than some others. For example, there has been less work on efficient algorithms for supporting powerful physical algebras over spatio-temporal data, or on efficient storage structures for describing how spatial data changes over time. Any comprehensive development activity in spatio-temporal databases is likely to yield new research results in these areas.

Query Processor Given that spatio-temporal queries are likely, on average, to be computationally demanding, operating as they do over complex data types and potentially large volumes of data, it is important that query optimisers can be developed that take full account of the most effective algorithms and indexing methods available. Research in this area is likely to give rise to new transformations on spatio-temporal queries, and both cost models and evaluation activities may identify needs for changes to index structures, algorithms or storage structures.

A further feature of the consolidation approach that seems important to the development of the field is the use of developed systems on real applications. The lack of experience with systems in practice leads to a minimal feedback loop, in which system limitations drive research on the basis of real needs. Research on the generation of spatio-temporal data sets for benchmarking purposes is useful (e.g., [7]), but there is a need not only for comprehensive benchmarks but also for experience reports with challenging applications.

Acknowledgements: Spatio-Temporal database research at Manchester is supported by the UK Engineering and Physical Sciences Research Council, whose support we are pleased to acknowledge. We are also grateful to our coworkers in this area at Keele for useful discussions, etc, namely: Mike Worboys, John Stell, Chris Johnson and Bo Huang.

References

[1] R. Cattell et al. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.

[2] D. Dey, T. Barron, and V. Storey. A complete temporal relational algebra. *VLDB Journal*, 5(3):167–180, 1996.

[3] M. Erwig, R. Guting, M. Schneider, and M. Vazirigiannis. Abstract and Discrete Modeling of Spatio-Temporal Data Types. *Geoinformatica*, 3(3):269–296, 1999.

[4] M. Erwig and M. Schneider. Developments in Spatio-Temporal Query Languages. In A. M. Tjoa et al., editors, *Proc. 10th Int. Workshop on Database and Expert Systems Applications*, pages 434–440. IEEE Press, 1999.

[5] R. Guting and M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, 4(2):243–286, 1995.

[6] T. Theodoridis, T. Sellis, and A. Papadopoulos. Specifications for Efficient Indexing in Spatiotemporal Databases. In *Proc. 10th SSDBM*, pages 123–132. IEEE Press, 1998.

[7] T. Theodoridis, R. Silva, and M. Nascimento. On the Generation of Spatiotemporal Datasets. In *Proc. 6th Int. Symposium on Spatial Databases*, pages 147–164. Springer Verlag, 1999.