# A Spatio-temporal Database System Based on TimeDB and Oracle Spatial

Alexandre Carvalho, Cristina Ribeiro, and A. Augusto Sousa
FEUP / INESC Porto
Rua Dr. Roberto Frias, 378
4200-465 Porto, PORTUGAL
+351 22 508 1827
{avc,mcr,aas}@fe.up.pt,
WWW home page: http://www.fe.up.pt

**Abstract**. The importance of the spatial component of data items has been long recognized and gave rise to a successful line of research and development in Geographic Information Systems (GIS). In many application domains it is also essential to deal with the evolution of data along time and to integrate spatial, temporal and other aspects of the information domain in an expressive and operationally effective manner. Until recently, temporal solutions provided by spatial database systems were semi-temporal approaches lacking full temporal support. As a consequence, most spatial database systems manage snapshots of the present state of facts without fully exploiting historical temporal aspects. This paper provides preliminary results on a spatiotemporal database implementation. The proposed system builds on existing database technologies, TimeDB and Oracle Spatial, for temporal and spatial support, respectively. The justification for the choice of these technologies is given, based on the state of the art in spatial and temporal database research. The integration of the spatial and temporal components is achieved with the extension of the TimeDB implementation layer. A set of goals has been established in order to cover both the integration of the spatial support and the enforcement of the temporal requirements in the extended system. Issues and solutions are presented and illustrative examples show the use of the implemented functionalities.

## 1 Introduction

Traditional databases model and keep information about some part of a real or artificial domain. Regarding the temporal aspects of facts that occur in the real word, these databases allow capturing some essence of time, which generally consist in a snapshot view of the world limited to the last update. This limitation becomes critical

*Please use the following format when citing this chapter:*

Carvalho, A., Ribeiro, C., Sousa, A., A., 2006, in International Federation for Information Processing, Volume 205, Research and Practical Issues of Enterprise Information Systems, eds. Tjoa, A.M., Xu, L., Chaudhry, S., (Boston:Springer), pp.11-20.

when there is the need to capture the evolution of facts over the time. To overcome this limitation, that is, to manage temporal aspects of facts in databases, Date [0] considers two distinct approaches: the semi-temporal approach, where the representation of historical data is done with timestamps, and the full-temporal approach, where the database must record the time when a fact is current in the reality - represented as an interval or a period - and in some application domains, must keep record of the time when a fact is current in the database. Such times are respectively called valid-time [0, 0] and transaction-time [3, 4]. If the first approach may lead to severe difficulties in handling some constraints and queries [0] the fact is that the full temporal approach has a higher level of complexity, requiring additional functionalities in the database system such as temporal data types, temporal operators (like begin, before, meets, contains, overlaps and coalesce) and temporal functions [0]. On the other hand, if most of the application domains are temporal by nature [0], it is also factual that some of these domains also contain spatial information [0]. In such situations it is of critical importance to manage seamlessly integrated temporal and spatial aspects. Examples of spatiotemporal domains are all the critical traffic management domains, where commercial navy, trains and airplane vessels must be permanently tracked. In the past few years spatiotemporal research has taken several important steps, but there are few implementations that address the problem of time and space in depth. Currently there are still many solutions following the so called semi-temporal approach, where much of the processing of the temporal aspects of temporal data is done at the application layer - increasing data and program complexity [0] - and not by the underlying DBMS layer, due to the lack of full temporal support. In an effort to overcome these limitations, the present work concerns the integration of two well known database technologies – TimeDB [0] and Oracle Spatial [0, 0] – in order to provide general spatial and temporal support without reducing the functionality provided by both technologies when working separately, through the ability to execute ATSQL2 [0] statements combined with spatial operators and functions. Our motivation to develop this integration is the creation of a full spatiotemporal relational database system (with bitemporal support) that constitutes the underlying layer of a larger system dedicated to manage present and past spatiotemporal urban data. In the next section we explain the selection of TimeDB for temporal support and Oracle Spatial for spatial support. In section 3 we present the reasons why the two technologies provide good integration for creating a spatiotemporal database prototype system and we point out the goals in joining the two database technologies as well as the issues that must be addressed in order to do so. Section 4 makes an overview of the system and its modules. Section 5 describes the main changes required in the translation algorithm in order that the translation process generates valid snapshot and spatial SQL. Section 5 concerns the results obtained, regarding the temporal and spatial goals. Finally, in section 7 we summarize the main conclusions of this paper and point out directions for current and future work.

## 2 Temporal and Spatial DBMS Implementations

Griffiths [0] states that the considerable design and implementation effort required to develop complete full spatio-temporal database systems is the main reason why research on temporal databases has focused on specific subparts of the problem like indexing or join algorithms and, consequently, there are so few implementations. The survey in [0] on temporal database systems classifies several systems - ARCADIA, Calanda, ChronoLog, HDBMS, TDBMS, TempCASE, TempIS, TimeDB, TimeIT, TimeMultiCal, and others - according to carefully thought-out criteria grouped in families. This author points out important conclusions on the tested implementations, like the dominance of the relational model with timestamped tuples and the fact that valid-time dimension has been the focus of attention leaving transaction time to a second place. More recently TEMPOS [0] and Tripod [0] emerged as temporal extensions of the ODMG object model, trying to overcome important limitations of previous extensions (TAU, TOOBIS, T_ODMG). TimeDB [0] is a client-side system [0] implementation that uses ATSQL2, provides bitemporal statements, supports upward compatibility and temporal upward compatibility, allowing legacy data and code to maintain the usability [0]. Being a layered [0] system, TimeDB can manage information from distinct DBMS technologies, among which is Oracle DMBS. In our opinion, although a client-side temporal implementation, TimeDB is not penalized by limited temporal functionality thanks to the way TimeDB deals both with translation of temporal statements and with management of temporary results. Steiner [0] considers that the translation algorithm of TimeDB can be used to translate different temporal query and modification languages into standard SQL statements. The execution of the resulting statements stores temporary results that are produced by a statement and consumed by the next one, avoiding collecting intermediate results and corresponding performance problems. Concerning spatial DMBS implementations, Medeiros [0] presents a survey focused on databases for GIS discussing several design criteria, such as data models, spatial operators, relationships, query modalities and optimization, data storage and access methods. Over these design issues, Medeiros refers database technologies and relates them to each design criteria. According to [0] a spatial DBMS must comply with three requirements: (a) to be a DBMS; (b) to provide spatial data types both for the definition of the data model and the manipulation by the query language, (c) to provide efficient algorithms for spatial operations like spatial *join* or multi-dimensional access methods (indexing). Having this in mind three Spatial DBMS implementations can be presented: ESRI ArcSDE, PostGIS and Oracle Spatial. The first constitutes a layered implementation that enhances non-spatial DBMS with the spatial support described. The other, PostGIS and Oracle Spatial [0][0] provide spatial support directly by the DBMS kernel through a spatial schema, a spatial indexing mechanism, a set of operators and functions for area-of-interest queries, spatial join queries and spatial analysis operations. Together, these functionalities allow the storage, retrieval, update and query of collections of spatial features.

## 3 Temporal and Spatial DBMS Implementations

The selection of Oracle Spatial as the support for spatial data in the proposed spatio-temporal system results from its support for multiple dimension geometry, the usage of spatial operators and spatial indexes, and a query language that is an extension to SQL and whose constructs can provide good integration with TimeDB ATSQL2, without harming temporal and spatial semantics. The integration of such technologies also permits the modified TimeDB layer to deal with just one DBMS for both temporal and spatial domains. The scenario of having two different DBMS managed by the modified TimeDB implementation would represent a critical performance, due to having to transfer temporary results between two DBMS. This approach would challenge one of TimeDB original major benefits concerning performance, namely the avoidance of intermediate results storage by the layer. Having settled on the underlying technologies we proceed to address the goals we expect to achieve with such integration, namely spatial related goals, temporal related goals and usability goals. Spatial related goals consist in providing TimeDB with the ability to deal with Oracle Spatial schema, geometry data types, spatial (and aggregate) functions and spatial indexes during the parsing, translation to non-temporal statements and evaluation processes, without impairing spatial functionality. Such ability provides that spatial functions and operators can be used within the modified ATSQL syntax, in order to provide spatial projections and selections. Such goals require functionalities that are not present in the original TimeDB implementation like being able to address the *mdsys* schema, in order to access and manipulate data and structures like geometry metadata tables (*mdsys.user_sdo_index_metadata*) and index metadata (*user_sdo_geom_metadata*). Also, the fact that TimeDB only addresses the tables and columns that are registered in its own metadata tables requires the proper insertion, on such tables, of the description of each column of the previous geometry metadata tables. Another issue concerns the use of spatial data types, for example *mdsys.sdo_geometry* that is widely used in the spatial domain. Among the spatial data types the aggregate data types, for example, *mdsys.sdoaggrtype* provides support for spatial aggregate functions like *sdo_aggr_union*. Finally, special care must be also provided for data type object methods, for example, *get_gtype*, from *mdsys.sdo_geometry* data type. Solving such issues provide the support for statements like:

*create table countries (name char(20), capital char(20),* ***boundary mdsys.sdo_geometry****) as validtime*

*validtime period [1100-1350)*
*select* ***sdo_aggr_union (mdsys.sdoaggrtype****(c.boundary, 0.005)) from countries c;*

Modifications are also required in order to be able to create and delete spatial indexes, through proper SQL statements, since indexation provide accelerated access method that is required for using spatial functions like *sdo_filter* or *sdo_within_distance*. Finally, concerning spatial related goals, it is common for a spatial data type, function or operator to have several spatial arguments, each one being another data type or spatial function with its own arguments. The support for parsing, translating and evaluating inner arguments must also be added in the

modified TimeDB layer. The following SQL insert statement provides an illustration of inner arguments situation:

*nonsequenced validtime period [1250-1590) insert into countries values ('portugal', 'lisboa',* ***mdsys.sdo_geometry (2003, null, null, mdsys.sdo_elem_info_array (1,1003,1),...,80.0,607.0))****);*

Special attention must be given to column references that are used as spatial arguments since the translation algorithm of TimeDB substitutes table and column aliases with temporary references during the translation process [0]. For example, in the following statement column references to boundary and name from table aliases a and b are replaced by internal representations of TimeDB:

*validtime select a.name from countries* ***a****, countries* ***b*** *where a.name = 'portugal' and b.name = 'poland' and sdo_geom.sdo_distance(****a.boundary, b.boundary****, 0.005) <=0;*

On the other hand, the temporal related goals expected to achieve with the proposed spatio-temporal integration are the goals already pointed out by Steiner [0], briefly summarized as "upward compatibility, temporal upward compatibility and orthogonality on valid time and transaction time together with the requirements listed in the definition of temporal completeness (syntactical similarity, sequenced and non-sequenced semantics of statements, substitutability of a relation in a query by another query and the support of temporal comparison predicates for time intervals [0])". This means that we expect the temporal functionality provided in original TimeDB to remain unharmed, as presented in section 6.

## 4 System Overview

The integration of spatial data management presented in the previous section within the original TimeDB implementation carries modifications to all TimeDB core modules (Fig. 1).

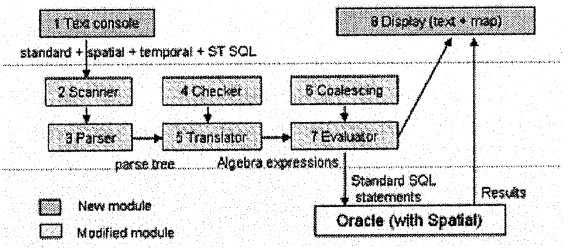


**Fig. 1:** modified architecture of TimeDB

In order to do so, the implementation of the proposed system relies both in changing and overriding original functionality. For example, the Scanner module has many changes, in order to recognize the spatial constructs. Also the Parser module, responsible for parsing and collecting metadata of SQL statements has several

changes in order to support - among other functionality - Oracle spatial tables, method calls in spatial attributes, spatial indexing, spatial functions and operators, spatial arguments, with the possibility of each argument being another function with arguments. Furthermore, in order to keep information about these new features, we has to reformulate several structures, since the parsing process results in a binary parse tree of scalar expression objects, each of them having the required metadata to create the equivalent spatial and non-temporal SQL statements. This metadata is used by the Translator module, responsible for mapping temporal algebra into snapshot equivalent algebra. This module has also several changes in order to accommodate the proper management of spatial columns (Section 5) by temporal operators and the correct translation of spatial projection and spatial selection operators.

## 5 Changes to the Translation Algorithm

According to Steiner [0], the algorithm for the translation of temporal queries into non-temporal standard SQL queries translates temporal statements into temporal algebra expressions using temporal set operators (union, intersection, difference), where each argument to one of these algebra expressions is a simple temporal algebra expression or the result of another temporal set operator. The integration of Oracle Spatial functions and operations within the translation algorithm does not amount to a major problem since spatial projection and spatial selection operations are, from the perspective of temporal algebra, regarded as simple non-temporal projection and selection and operation. Spatial data types are treated by the translation algorithm, concerning temporal algebra, as other non-temporal data types like number or character strings. Nevertheless column references used as spatial arguments require proper dealing. Also, changes must be performed over the translation algorithm in order to allow temporal set operations and the valid-time coalescing to perform correctly. In both situations, the issue is raised when evaluating SQL code (generated by the translation algorithm) that contains selection operators which compare values of distinct tupples, concerning the same spatial column. Since Oracle Spatial cannot compare two spatial values through the equal operator the SQL code becomes invalid. To overcome this limitation our proposal is to go through the snapshot equivalent mappings for temporal set operators and temporal coalescing, and use:

$sdo\_geom.relate(table_1.column_a,$ *'equal'*, $table_2.column_a,$ *<a spatial tolerance>)* = *'equal'*

when comparing values in a geometry column, instead of

$table_1.column_a = table_2.column_a$

Providing these changes in the temporal intersection set operator, the temporal difference set operator and the unitemporal coalescing operator overcomes the problems identified in the snapshot equivalent SQL code (since through *sdo_geom.relate*, Oracle Spatial provides a spatial comparison between the spatial values).

# 6 Known Results

Snodgrass [0] and Steiner [0] consider that upward compatibility and temporal upward compatibility are ATSQL2 requirements related with database migration. In our prototype system, upward compatibility is maintained through TimeDB original functionality, but regarded from the spatio-temporal perspective: any legal ATSQL2 statement has the same semantics and validity as in the merged ATSQL2 and Oracle Spatial syntax. An important requirement states that each legal SQL query and modification statement, executed on a temporal database, leads to the same result as if it were executed on the corresponding non-temporal database [0][0]. This requirement is also maintained valid with Oracle Spatial legal SQL query and modification statements. Also, the two classes of temporal statements in ATSQL2 (sequenced and non-sequenced) have their semantics unmodified, after the inclusion of Oracle Spatial components. To test that TimeDB functionality has been left unharmed we tested and compared the results of the demos included with original TimeDB with the results obtained in a TimeDB unmodified implementation, leading to the conclusion that, in the subset of temporal valid-time domain, now integrated with spatial support, we have not introduced any limitations or made any simplifications. Concerning spatio-temporal domain, the following sub-sections address the results of temporal compatibility and semantics, from the perspective of temporal and spatial seamless integration.

## 6.1 Upward Compatible queries and Temporal Upward Compatible queries

Upward compatible queries [0, 0] protect investments of legacy code and provides a gradual process of migration to a temporal DMBS of legacy code and data [0], the proposed system provides that legacy Oracle Spatial code and data can still be used providing results equivalent to those obtained in plain Oracle Spatial DBMS, that is, without awareness of the temporal support provided. Regarding temporal upward compatible queries, they allow the upgrade of legacy applications to temporal database systems and the invariance of the semantics and functionality of legacy statements [0]. This means that snapshot spatial queries issued over valid-time spatial tables will retrieve snapshot spatial results concerning the current valid-time state. For example, providing a valid-time table countries4, the following statement:

```
select * from countries4
where sdo_geom.sdo_area(countries4.boundary, 0.005) > 5000;
```

retrieves a snapshot result (polygonal geometry). The tupples used by this query are the subset that is considered valid at the time of execution: the non-temporal SQL issued by the evaluator module contains two selection operations that make possible to ignore other tupples (64416774529 is the chronon when the statement was created in the Translator module):

```
select alias_tdb0.name as name, alias_tdb0.capital as capital,
alias_tdb0.boundary as boundary from countries4 alias_tdb0 where
sdo_geom.sdo_area(alias_tdb0.boundary, 0.005) > 5000 and
```

*alias_tdb0.vts_timedb <= **64416774529** and alias_tdb0.vte_timedb > **64416774529***

## 6.2 Sequenced and Non-sequenced Queries

In what concerns sequenced queries, good for providing historical results [0, 0], temporal logic is applied and operations interpret the timestamps of tuples and use this interpretation for the calculation of the resulting tuples timestamps. TimeDB uses temporal logic to calculate the resulting relation for a query which has, for each tuple, a valid-time timestamp value associated. In such queries, spatial projection and selection operations are regarded as non-temporal operations, like the ones acting on numerical and alphanumerical columns. For example, considering a relation instance that contains historical spatio-temporal information (name, capital and boundary) illustrated in Fig. 2, the following sequenced query retrieves the valid-time periods and respective spatial union for all the countries boundaries. The spatial results of executing the statement are displayed in Fig. 3.

*validtime period [1100-1350)*
*select sdo_aggr_union(mdsys.sdoaggrtype(c.boundary, 0.005)) from countries c;*

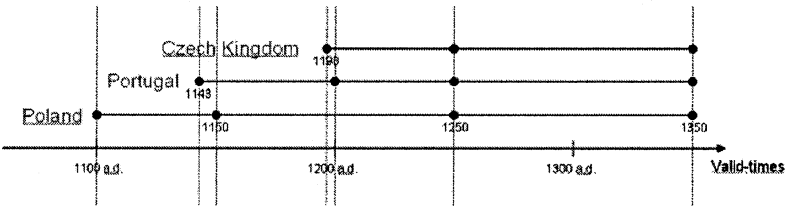


**Fig. 2:** Schema of database facts regarding Poland, Portugal, and Czech Kingdom, from 1100 A.D. to 1350 A.D.

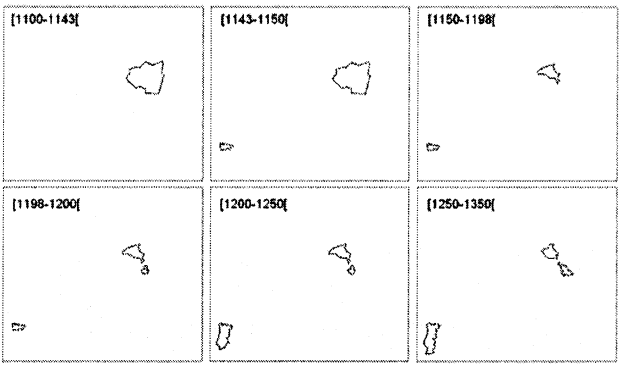


**Fig. 3:** Spatiotemporal results from the sequenced valid time spatial union query.

Concerning non-sequenced queries, which use non-temporal logic, attributes of temporal data type, like valid time start and end, are managed as other numerical attributes. Eventually, all database states can be used to calculate the resulting

relation, as opposed to what happens in sequenced queries. For example, the following query retrieves the union of all country's boundaries disregarding the valid-time of tupples:

*nonsequenced validtime*
*select sdo_aggr_union(mdsys.sdoaggrtype(c.boundary, 0.005)) from countries c;*

## 7 Conclusions and Ongoing Work

Spatiotemporal database technology is required for properly addressing the subset of temporal applications that also manage spatial data and, currently, has many domains of application, from traffic management to urban data management to enterprise information management. As there are only a few systems satisfying both temporal and spatial aspects the presented work intends to contribute to this area by proposing a spatiotemporal database system based on two database technologies: TimeDB and Oracle Spatial. The selection of TimeDB is due to the amount of temporal support provided, as compared to other implementations. Oracle Spatial choice is a consequence of selecting TimeDB, since we want the spatiotemporal layer to deal with only one underlying DBMS. This also maintains the layer free from retrieving temporary results, which would harm overall performance. Regarding the integration issues, the proposed spatiotemporal database system has taken into consideration several groups of functionalities. At the parsing and interpretation level, we have identified the enhancement required on TimeDB scanner and parser modules in order to allow parsing and interpretation of Oracle spatial data types, spatial indexing, spatial functions (and aggregate functions) and spatial operators. At the level of translating temporal SQL to equivalent snapshot SQL statements, we have modified the translation algorithm in what concerns the inclusion of spatial algebra expressions that are integrated in the translated non-temporal, spatial, SQL. In this translation module we have also dealt with the issue of translating column references to the correct table and column aliases. These changes made possible for temporal operations to perform correctly (like unitemporal coalescing). Finally, at the level of retrieval of results, we have created new functionality that goes beyond textual results, being aware of spatial results and, consequently, displays them according to their spatial data type. Concerning the temporal goals, we have concluded that the integration of Oracle Spatial constructs with TimeDB ATSQL does not compromise the original temporal support of TimeDB. Upward compatibility, temporal upward compatibility, sequenced and non-sequenced semantics on valid time are maintained unharmed. Also this integration of both technologies has no impact on spatial functions: although issued in a spatiotemporal context the spatial support maintains the original functionality. Thanks to this fact and to the ability to process the most common spatial data types, all spatial operators and the great majority of spatial functions we can state that spatial goals proposed have been accomplished. The correctness of the implemented functionalities has been tested on a small data set, built to illustrate the main spatial and temporal features and dependencies. Current work includes the use of the prototype system in the domain of urban planning, where extensive geo-referenced data sets exist and querying the temporal

components of information is still a challenge. Ongoing work includes providing and testing transaction time support and bi-temporal support integrated with spatial support.

## References

1. C. Date, An Introduction to Database Systems, 7th edition (Addison-Wesley Chapter 22, 2000), pp. 730-768.
2. R. Snodgrass, M. Böhlen, C. Jensen, A. Steiner, Adding Valid Time to SQL/Temporal, SQL/Temporal Change Proposal, ANSI X3H2-96-501r2, ISO/IEC JTC1/SC21/WG3 DBL MAD-146r2, 1996.
3. C. Jensen, Temporal Database Management, PhD, 2000.
4. R. Snodgrass, M. Böhlen, C. Jensen, A. Steiner, Adding Transaction time to SQL/Temporal, SQL/Temporal Change Proposal, ANSI X3H2-96-502r2, ISO/IEC JTC1/SC21/WG3 DBL MAD-147r2, 1996.
5. A. Steiner, A Generalization Approach to temporal Data Models and Their Implementations, PhD, Zurich, 1998.
6. R. Bastos, Autonomic Computing Approach for Resource Allocation, *Expert Systems with Applications* **28**, 9-19 (2005).
7. M. Dumas, C. Fauvet, P. Scholl, TEMPOS: A Temporal Database Model Seamless Extending ODMG, LSR-IMAG, University of Grenoble, PhD, 2000.
8. Oracle Corporation, Oracle Spatial 9.0.1 - User's Guide and Reference, June 2001.
Oracle Corporation, Oracle 9i, Application Developer's Guide - Object-Relational Features, March 2002.
9. M. Böhlen, C. Jensen, R. Snodgrass, Evaluating the completeness of TSQL2, Recent Advances in Temporal Databases, J. Clifford and A. Tuzhilin, 1995.
10. T. Griffiths, A. Fernandes, N. Paton , S. Jeong, N. Djafri, Tripod: A Spatio-Historical Object Database System, Mining Spatio-Temporal Information Systems, The Kluwer International Series in Engineering and Computer Science: Volume 699, 2002, pp. 127-146.
11. M. Böhlen, Temporal Database System Implementations, *SIGMOD Record* **24**(4), (1995).
12. C. Medeiros, F. Pires, Databases for GIS, *SIGMOD* **23**(1), (1994).
13. C. Vassilakis, P. Georgiadis, A. Sotiropoulou, A Comparative Study of Temporal DBMS Architectures, DEXA, 1996, pp. 153-164.
14. K. Torp, C. Jensen, M. Böhlen, Layered Implementation of Temporal DMBSs – Concepts and Techniques, Time Center Technical Report, 1997.
15. J. Allen, Maintaining Knowledge about Temporal Intervals, *Communications of the ACM* **16**(11), 832-843 (1983).