Spatial TimeDB – Valid Time Support in Spatial DBMS

Alexandre Carvalho, Cristina Ribeiro, A. Augusto Sousa
FEUP, INESC Porto
{avc, mcr, aas}@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). Unfortunately regular GIS do not support the temporal dimension of urban data in a convenient manner, mainly because priority has been given to spatial unresolved issues. Temporal snapshot approaches, in which the most recent state is kept, dominate the area of urban data management. Other approaches manage the temporal aspects of data at the application level, disregarding the state of the art in temporal database management. To our knowledge, no current GIS provide full temporal support over the valid time domain, thus compromising the integrated management of spatio-temporal data at the DBMS level. In an effort to overcome some of these limitations, the present paper proposes the integration of the spatial extension of a well known spatial DBMS (Oracle Spatial) within the ATSQL2 language. We expect that such integration provides spatio-temporal support without harming any constructs or semantics defined in ATSQL2. A prototype system implementing the integrated language is presented. Examples are also given in order to illustrate the extended functionalities.

1 Introduction

Spatio-temporal data management is a critical problem in several application domains [1][2] like facility management, environmental systems [3], transports and urban data management, from government level to local municipalities. Regarding spatial data management, most of these application domains rely on mature Geographical Information Systems (GIS). Considering the temporal approach in these systems two alternatives are followed: to manage the most recent state for spatial entities and to timestamp spatial entities by providing date/time simple types. The second approach, designated by Date, as semi-temporal, leads to severe difficulties in handling some constraints and queries [4]. Due to the lack of adequate temporal support in the GIS, some questions involving time can be complex (but possible) to express in standard SQL [4]; for example, consider the following questions. In the last 50 years and in some specific area where did terrain division occur? How did street X developed over the last 10 years? What were the gas infrastructure interventions from company X on city block Y?

Currently, in GIS, most of the processing of the temporal aspects of data is done at the application layer, increasing data and program complexity [5]. Due to the increasing demands of representing historical data and to the possibility of querying DBMS with questions involving distinct temporal semantics [6], there is an increasing requirement of temporal data management in GIS, thus providing increased SQL expressivity and relieving the application layer of handling both spatial and temporal semantics.

In an effort to contribute to temporal data management in GIS the present work proposes adding a spatial extension to ATSQL2 [7] in order to provide spatio-temporal data management, through the ability to query the underlying DBMS with questions having sequenced and non-sequenced valid-time semantics, integrated with the usage of spatial data types, operators and spatial functions. Such statements are expressed in a SQL that results from integrating Oracle Spatial SQL within ATSQL2.
2 Managing spatial and temporal data seamlessly integrated

Based on the analysis of Oracle Spatial SQL and ATSQL2 languages our proposal for extension is based on the characteristics that:

- ATSQL2 is a strict superset of SQL, which means that this language only adds new constructs and semantics [6] without compromising any of regular SQL functionality;
- Oracle Spatial SQL [8] language is also a strict superset, regarding SQL.

The previous characteristics permit regular SQL to maintain its functionality in the proposed extension. On the other hand, temporal semantics, defined in ATSQL2 by temporal predicates [6], determine which algebra to use: sequenced semantics imply temporal algebra while other semantics imply snapshot (non-temporal) algebra.

In the context of temporal algebra, spatial operators and spatial functions can be translated to simple projection and selection algebra operations, by using the mappings proposed by Snodgrass [6], thus allowing the translation to regular snapshot-equivalent SQL. Based on these premises, the temporal dimension can be managed orthogonally to the spatial dimensions, providing that first we determine which semantic is being used and translate the spatial functions and operations (as any other selection and projection algebra operations) according to the respective snapshot equivalent mappings. Consequently, the temporal dimension prevails over the spatial dimensions, meaning that each statement can start with a temporal predicate in order to trigger the usage of distinct semantics.

The result is a spatio-temporal SQL language that maintains temporal support for upward compatibility, temporal upward compatibility, syntactical similarity, sequenced and non-sequenced semantics of statements and the support of temporal comparison predicates for time intervals [5]. It also accommodates proper spatial data management, through spatial data types, functions and operators and spatial indexing mechanism. Statements issued can combine temporal and spatial constructs.

3 Proof of concept

In order to provide a proof of concept for the proposed spatio-temporal SQL we have analyzed several temporal implementations of ATSQL, among which TimeDB [9] seemed one of the soundest prototypes available. It was also taken into consideration the fact that TimeDB could use Oracle (not spatial) as an underlying DBMS. Having this in mind, our next step was to analyze the TimeDB architecture, and identify the modifications needed for the ATSQL2 to accommodate the spatial syntax constructs, namely:

- Scan and parse spatial data types (and aggregate types), functions and operators;
- Interpret arguments within spatial functions and spatial operators with several levels of depth.
- Provide correct translation of spatio-temporal SQL statements to standard SQL with spatial extension, by maintaining SQL mappings [6], but proper dealing with spatial constrains (managed orthogonally regarding time);
- Finally, issue the translated SQL statements to the underlying DBMS and collect results from it. Results maintain the possibility of being snapshot or valid-time results. The new thing is that spatial attributes can also be retrieved, provided they are referred in the projection list of spatio-temporal queries.

The proposed system requires modifications in all of original TimeDB modules, shown in grey in Figure 1.

![Figure 1: spatio-temporal layer architecture](image)

For example:
- the Scanner having the ability to recognize the new spatial constructs or maintaining the
character case, in string tokens, which are required by Oracle Spatial;
- The Parser module having the ability to support, among other features, spatial tables, method calls in spatial attributes, spatial indexing, spatial functions and operators and spatial arguments, with the possibility of each argument being another function with its own arguments. The result of such changes are binary parse trees, filled with spatio-temporal metadata, which serves as input to the Translator module where snapshot-equivalent mappings transform the tree into a sequence of snapshot, spatial SQL statements.
- The Translator module being able to process relation attributes or aliases used as spatial arguments, which require proper snapshot-equivalent mapping in valid time sequenced semantics context, and unitemporal coalescing of relations that contain spatial attributes.

4 Spatial data management in the translation process

According to Steiner [9], the basic idea used in the translation algorithm is that a temporal query is translated into temporal algebra expressions using the temporal set operations (temporal union, temporal intersect and temporal difference). Each argument to these expressions is either a simple algebra expression (temporal select, temporal project, temporal cross product) or the result of another temporal set operation. These temporal algebra operators have snapshot equivalents [6], which allow the mapping to regular algebra and to SQL. Based on these assumptions, the spatial projection and spatial selection correspond, in temporal algebra, to simple temporal algebra operations, due to orthogonality of temporal and spatial dimensions. Hence, spatio-temporal queries are translated to Oracle Spatial SQL through the same snapshot equivalence mappings.

Considering our snapshot-equivalent mapping process there are however a few issues to address in order for the temporal set operations and the valid-time coalescing to work correctly: spatial comparison functions are required when comparing two spatial attributes, instead of the equal comparison operator. For example, the following statement illustrates a part of the mapping of temporal set difference to regular SQL:

```
insert into r1
select a0.vts_#5, a1.vts_#5, a0.a1, ..., a0.an
from r1 a0, r2 a1
where a1.vts_#5 > a0.vts_#5 and
  a1.vts_#5 < a0.vts_#5 and
  a0.a1 = a1.b1 and
  ... and
  a0.an = a1.bn;
```

In the previous statement, the comparison between values from different tuples regarding the same attribute in table R1 and in table R2 (for example, a0.a1 = a1.b1) cannot be done through the usage of the equal operator, if a1 and b1 are spatial attributes. Hence, considering the snapshot equivalent mappings, for every comparison between two spatial attributes, we use Oracle Spatial function `sdo_geom.relate`. For example, the comparison `a0.a1 = a1.b1` is modified to:

```
sdo_geom.relate(a0.a1, 'equal', a1.b1, <a spatial tolerance>) = 'equal'
```

5 Tests and results

To test that the referred temporal characteristics of ATSQL2 were left unharmed we ran the demos (concerning snapshot and valid-time support) included with the release of TimeDB 2.0 and compared the results to the ones obtained in our prototype system. We concluded that the integration of spatial support within the temporal support did not introduce any limitations to the temporal support provided in TimeDB 2.0.

The proposed spatio-temporal prototype maintains upward compatibility and also permits that any legacy data and code (including the spatial domain) can still be used after the migration to our prototype system. On the other hand, conventional Oracle Spatial SQL statements, unaware of the temporal extensions, maintain their functionality unharmed, since, in this situation, the statements issued by our prototype system concern only the present state of the spatio-temporal database. Also, temporal sequenced and non-sequenced statements maintain their semantics unaltered, after the
inclusion of support to Oracle Spatial data types, operators and functions. As an example, consider
data presented in Figure 2, where each valid-time
tuple is represented as a horizontal segment
limited by two bullets (from A.D. 1100 to A.D. 1350).

![Figure 2: Schema of database facts regarding Poland, Portugal, and Czech Kingdom, from A.D.
1100 to A.D. 1350.](image)

Taking these facts into consideration the following statement:

```sql
validtime period (1100-1350) select
sdo_aggr_union(mdsys.sdoagrtype(c.boundary, 0.005)) from countries c;
```

returns six spatial unions with corresponding
valid time periods, represented in Figure 2 as the
vertical lines. Figure 3 illustrates the spatial
results that are retrieved from this sequenced
valid time query.

![Figure 3: Spatiotemporal results from the sequenced valid time spatial union query.](image)

A non-sequenced valid time semantics example is
following query which retrieves the union of all
country’s boundaries that were closer than 300
spatial units from polish borders.

```sql
nonsequenced validtime select sdo_aggr_union (mdsys.sdoagrtype (c.boundary, 0.005))
from countries c
where exists (select * from countries d where
sdo_geom.sdo_area(d.boundary, 0.005) < 300 and d.name = 'poland');
```

This statement results in a single shape - a
multi-polygon, with no temporal period specified.

Unitemporal coalescing [10] with spatial
attributes and/or spatial constrains has also been
addressed. As an example, consider the following statement:

```sql
create view max_areas as validtime
select max(sdo_geom.sdo_area(c.boundary, 0.005)) as area from countries c;
```

That creates a view that retrieves, for each
period, the maximum area of countries existing at
that time.

If we also that we want to retrieve, for each
period, the name of the country that has the
largest area, one possible solution is to provide
the following statement:

```sql
validtime select a.name, a.area
from countries a, max_areas b
where sdo_geom.sdo_area(a.boundary, 0.005) = b.area and validtime(a) overlaps validtime(b);
```

The execution of this statement retrieves the
results displayed on the left side of Figure 4.

![Figure 4: (Left side) results of the previous query. (Right side) coalesced results over valid time.](image)

In this illustration there are several tuples with
equal non-temporal values. For example, tuples
with valid-time [1040-1143) and [1143-1150],
about Poland, contain the same area, which is
14371.5.

The coalesced form of the previous results can
be calculated by placing the coalescing operator
at the end of the previous statement:

```sql
validtime (select a.name, a.area
from countries a, max areas b
where sdo_geom.sdo_area(a.boundary, 0.005) = b.area
and validtime(a) overlaps validtime(b)) coalesce
```
thus providing results in the coalesced form, illustrated on the right side of Figure 4.

6 Conclusions and ongoing work

The present work intends to contribute to spatio-temporal data management by proposing the integration of a spatial extension within the ATSQL2 language. As presented, such integration is possible because ATSQL2 and the spatial extensions are both supersets of SQL and due to orthogonal treatment of time and space. SQL mappings proposed by Snodgrass maintain their validity even when there are spatial projections and selections involved.

Regarding our prototype system, used as proof of concept, we based it on TimeDB 2.0: it was subject to changes in all of its components in order to support spatial data types, functions and operators and to be able to use them during mappings of spatio-temporal SQL statements to spatio-snapshot SQL statements. Another important issue, during this translation, was to correctly deal with spatial columns and aliases used as spatial function arguments, within valid time sequenced semantics.

The correctness of the implemented functionalities has been tested both with the demos that come with TimeDB2.0 and with a small spatio-temporal dataset, which allowed us to conclude that the proposed spatio-temporal integration does not compromise the goals for upward compatibility, temporal upward compatibility, sequenced and non-sequenced semantics support on valid time. Also spatial support is fully available both in temporal and non-temporal contexts.

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.

7 References