Spatio-Temporal Indexing in Oracle: Issues and Challenges

Ravi Kanth V. Kothuri and Siva Ravada
Spatial Technologies, NEDC
Oracle Corporation, Nashua, NH 03062, USA
{Ravi.Kothuri, Siva.Ravada}@oracle.com

Abstract

Spatio-temporal indexing has become an increasingly important area of database research. In this paper, we describe how such data can be indexed using current features of Oracle Spatial such as Spatial indexing, Linear Referencing System, and Oracle's function-based indexes. We examine the issues and limitations of this approach and possible future enhancements. This approach could be used as a first guide to implement most spatio-temporal applications such as mobile-object tracking.

1 Introduction

With the rapid growth in the wireless industry and the advances in cellular technology, mobility is increasingly becoming an important aspect of data objects and queries. Consequently, the spatial database research community has recently proposed new data models, defined meaningful query semantics, and devised efficient search and access methods for non-static moving spatial data.

The motion is modeled along the temporal dimension. The notion of the ever-increasing current time poses a variety of challenges when time and space are combined. We may distinguish between two paradigms for research contributions: those that concern "historical" or predictable-route data, where the motion is more or less pre-determined, and those that concern "on-going-motion" data, where only the current velocity is known.

Since the location of an object needs to be updated after every time instant, on-going-motion data are usually modeled in parametric space in terms of the time and velocity of an object, which is assumed to be unchanging (and triggers an update whenever there is a change). Several proposals have been made in the research literature for indexing such data [4, 9, 11].

However, in most mobile applications, the start and destination of an object's motion can be pre-determined using the route computed in the in-car navigation system, and the location of the object at any time can be estimated roughly by using the speed limits on the route. In other applications, data capturing the past motion of an object is considered. Several research solutions [6, 10] have been proposed for such "historical" motion data.

In this paper, we describe how historical motion data can be indexed using existing technologies of Oracle. This solution should be reasonably efficient for most applications, and it extends readily the salient features of commercial databases, such as backup and recovery, concurrency, and replication, to spatio-temporal data.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2 Modeling Spatio-Temporal Data

We proceed to briefly describe the Oracle Spatial data model and how it supports linear referencing. Spatio-temporal data can be modeled as a direct application of linear-referenced spatial data.

Oracle Spatial models 2–4 dimensional spatial data using an *sdo_geometry* data type. For the 2-dimensional case, this data type models all the spatial data types defined by the Open GIS Consortium (OGC) and caters to most data occurring in GIS and CAD/CAM applications. The supported spatial data types includes simple primitive elements such as points, lines, curves, polygons (with and without holes), and complex elements that are made up of a combination of primitive elements. The sdo_geometry data type is implemented as an Oracle object datatype. This approach extends all the benefits of Oracle's object-relational database technology to spatial data, including replication.

In the above geometry model, Oracle Spatial allows numerical attributes to be associated with vertices of a linear feature (a line geometry). That is, in addition to storing a spatial (latitude, longitude) coordinate, each vertex can also contain two other numerical attributes referred to as the *measure* attributes. Some transportation customers use these measure attributes to represent the mile-post markers, traffic conditions, or speed-limits when the line string data represents a road network. Whereas indexing is supported on the 2-dimensional spatial coordinate data, linear referencing system (LRS) functions operate on the measure attributes of the geometry data. Some LRS functions are listed below. This linear-referencing system combines powerful spatial indexing capability with functional operations on linear measures and has been very useful in the transportation and other related industries.

- start_measure(g) and end_measure(g): return the starting and ending measure of an LRS geometry segment, respectively.
- clip_geometry(g, t) (or dynamic segmentation): clips a geometry segment at the specified interval of measures.
- split_geometry(g, measure): splits a geometry into two segments at specified measure value.

The linear referencing system can be used to support spatio-temporal data in Oracle.

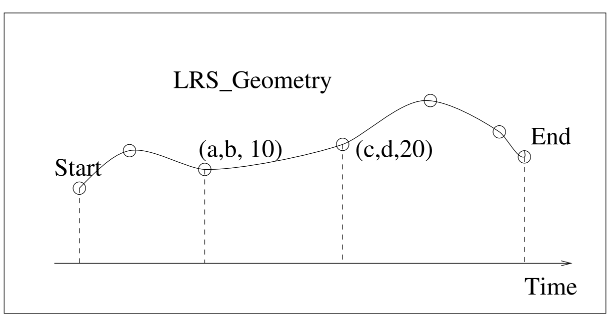

Figure 1: Example of a Spatio-Temporal Geometry Modeled Using an LRS_geometry

For moving data points, the line string geometry can model the trajectory of the object and one of the measure attributes could represent the temporal dimension. Figure 1 shows an example. The first two dimensions denote the (x, y)-dimensions, and the third dimension denotes time. Identifying the trajectory of the object between times 10 and 20 can be performed by calling the "clipping" function of the spatial linear referencing system. This model can easily be modified to include the enhancements from partially-persistent R-trees [5] where a moving object's trajectory is modeled as different geometries.

3 Indexing Spatio-Temporal Data

Although spatio-temporal data can be stored with both the spatial and temporal dimensions together as a single object, there is no explicit indexing support for spatio-temporal data in Oracle Spatial. However, both spatial and temporal dimensions can be extracted from the data without replicating the data and indexed using separate indexes. This approach is described in Figure 2.

During spatial indexing, the spatial dimensions are implicitly extracted from the spatio-temporal geometry and used in indexing.

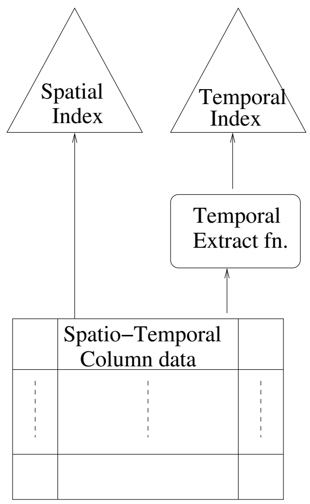

Figure 2: Indexing Spatio-Temporal Data in Oracle

The temporal index, on the other hand, is constructed by explicitly extracting the temporal aspect of the spatio-temporal geometry. This temporal aspect can be either a separate geometry, which is indexed by a 1-dimensional spatial R-tree index, or it can be the start and end measures of the geometries, which are indexed by "function-based" B-tree indexes. In either case, the function-based indexes in Oracle allow indexes to be constructed on functions of column values (in this case, functions returning the temporal aspect) without explicitly storing the function values. This means that the spatio-temporal geometry need not be redundantly stored/replicated, which helps ensure data integrity. Updates need only be performed on the spatio-temporal geometry column of the database table, and both (spatial and temporal) indexes are updated automatically by the database engine.

We procced to describe spatial indexes in more detail. Temporal data can be indexed similarly, using either spatial indexes or by B-tree indexes.

Oracle Spatial indexes 2-dimensional spatial data using either a quadtree or an R-tree. These indexes are implemented using the extensible indexing framework of Oracle [1, 7]. This framework allows for the creation of new domain-specific indexes and associated query operators and provides for the integration of user-specified query, update and index creation routines inside the Oracle server. To query the constructed spatial indexes, new SQL-level predicates, referred to as *operators*, are defined. These operators can be included in the "where" clause of a SQL statement to select data that satisfy a specified predicate with respect to a specified query window. Such operators are executed using index-associated procedures for query processing and allow for incremental processing of queries (see [1, 7, 8] for more details). Oracle Spatial provides the following query operators on 2-dimensional spatial data:

- sdo_filter query: returns geometries whose index approximations intersect those of the 2-d spatial query window.
- sdo_relate query: returns geometries that interact with a query window. Different types of interaction (most common being "intersects," and "inside"), based on the 9-intersection model [2, 3] can be specified. This operator is equivalent to the *ST_relate* operator of SQL/MM. (Most SQL/MM suggested methods are available in some form in Oracle Spatial.)
- sdo_within_distance (or within-distance) queries: identify geometries that are within a specified distance from the query geometry.
- sdo_nn (nearest-neighbor) queries: identify the k nearest neighbors for a specified query geometry.

Combined with the functions defined in the LRS to operate on the temporal dimension aspect of the geometry, these operators provide effective query capabilities on spatio-temporal data.

4 Spatio-Temporal Queries

A spatio-temporal query consists of a spatial query window (an arbitrary 2-d geometry) and a time interval. Alternately, a more complicated query in the form of an LRS geometry where different times are associated with different vertices, can be specified. However, this is not a common case, and we will discuss these limitations at the end.

Spatio-temporal queries can be categorized broadly into three categories [6]. First, in *spatial-range, temporal-range* queries, data geometries that satisfy both the spatial and temporal ranges are returned. Examples of such a query include identifying all cars that pass through a toll-booth during a specified time interval. Second, in *spatial-range, temporal-knn* queries, the k data geometries that satisfy the spatial range and are closest to the specified time interval are returned. Examples of such a query include identifying the most recent cars that passed through an accident site. Third, in *spatial-knn, temporal-range* queries, the k data geometries that satisfy the temporal range and are closest to the specified query window in spatial dimension are returned. Examples of such a query include identifying the closest restaurants for a moving object at the current time.

Next we describe how each of the above three query types could be answered using Oracle Spatial. The syntax of Oracle Spatial functions used next is simplified for clarity.

Spatio-Temporal Range Query This query is quite easy to implement using the spatial index operators and the measure functions of LRS. If q is the 2-d spatial window, and s, e delimit the time interval, the following query returns the ids of the result geometries. In the query, the sdo_relate operator retrieves the geometries that satisfy the query window q and compares their start and end measures with the specified time interval.

```
select gid from geom_tab a
where sdo_relate(a.geom, q, 'intersects')= 'TRUE'
  and sdo_lrs.start_measure(a.geom) <= e and sdo_lrs.end_measure(a.geom) >= s;
```

Spatial-Range Temporal-knn Query This query is also easy to implement using the spatial index operators and the measure functions of LRS as follows: First, geometries intersecting the spatial query window are retrieved from the database using the sdo_relate operator. Next, the intersecting portions of the resulting LRS data geometries are obtained using the sdo_intersection function. This function uses the query window q to return the intersecting portions of the resulting data geometries. Then the start and end measures are obtained from the intersecting LRS geometries. In the next step, using the start and end measures, the temporal-distance to the temporal query range is computed. This is possible by using a simple function such as, say, temporal_dist to compute the minimum distance between query and data geometry temporal end points. Finally, the resulting geometries are ordered based on the temporal distance, and the k nearest geometries are returned. The SQL query looks as follows:

```
select gid
from ( select gid,
         temporal_dist(s, e, start_measure(res), end_measure(res)) t_dist
       from ( select a.gid gid, sdo_intersection(a.geom, q) res
              from geom_tab a
              where sdo_relate(a.geom, q, 'intersects') = 'TRUE' )
       order by t_dist )
where rownum <= k;
```

Spatial-knn Temporal-Range Query This query is evaluated as follows: First, the temporal range is used to select a set of geometries. The resulting geometries are then clipped using the temporal range endpoints. Next, the spatial distance between the clipped geometries and the query geometry is computed using the sdo_distance function provided by Oracle Spatial. Then the geometries are ordered based on the computed distances of the clipped geometries to the query geometry. Finally, the k nearest data geometries in the above set are returned as the result. The SQL query looks as follows:

```
select gid
from ( select gid, sdo_distance(q, res) s_dist
       from ( select gid, clip_geometry(geom, s, e) res
              from geom_tab
              where sdo_relate(temporal_fn(geom),
                      geometry_constructor(s,e),'inside')='TRUE' )
       order by s_dist )
where rownum <= k;
```

5 Issues and Limitations

Although the presented approach allows for reasonably-efficient indexing of spatio-temporal data, several issues arise. Since only either a spatial or a temporal index is used in answering the queries, the performance may not be as effective as having a single combined, spatio-temporal index. Several performance-enhancing proposals exist that can be implemented in future releases. Next, several proposals also exist to segment each LRS geometry into multiple parts, thus reducing the MBR-area covered in spatial indexes. These are also likely to be implemented in future releases. Finally, the temporal dimension can only be modeled as a numerical attribute in the above model. Although not ideal, translation from date to numerical values can be performed in the application layer.

6 Conclusions and Future Work

In this paper, we described a model for storing and indexing spatio-temporal data in Oracle. This approach combines the spatial indexing, linear referencing, and function-based indexes of Oracle to support most types of queries on spatio-temporal data. As a consequence, advantages—such as concurrency, recovery, scalability, and replication—of storing spatio-temporal data in a commercial database are easily extended to spatio-temporal data. This approach marks the first step to indexing spatio-temporal data in Oracle using existing technology. Some of the limitations of this approach can be easily removed if both spatial and temporal attributes are treated together in a combined index and query processing mechanism. Also, indexing spatio-temporal data in parametric space may be necessary to cater to some mobile applications. These efficient indexing strategies will be explored in future releases.

References

[1] S. Defazio, A. Daoud, L. A. Smith, and J. Srinivasan. Integrating IR and RDBMS using cooperative indexing. In *ACM SIGIR Conf. on Information Retrieval*, pp. 84–92, 1995.

[2] M. J. Egenhofer. Reasoning about binary topological relations. In *Symp. on Spatial Databases*, pp. 271–289, 1991.

[3] M. J. Egenhofer, A. U. Frank, and J. P. Jackson. A topological data model for spatial databases. In *Symp. on Spatial Databases*, pp. 271–289, 1989.

[4] G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. In *ACM Symp. on Principles of Database Systems*, 1999.

[5] G. Kollios, V. Tsotras, D. Gunopulos, A. Delis, and M. Hadjieleftheriou. Indexing animated objects using spatiotemporal access methods. *IEEE Trans. on Knowledge and Data Engineering*, 13(5):758–777, 2001.

[6] K. Prokaew, I. Lazaridis, and S. Mehrotra. Querying mobile objects in spatio-temporal databases. In *Symp. on in Spatial and Temporal Databases*, pp. 59–78, 2001.

[7] K. V. R. Kanth, S. Ravada, J. Sharma, and J. Banerjee. Indexing medium-dimensionality data in Oracle. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 521–522, 1999.

[8] K. V. R. Kanth and S. Ravada. Efficient processing of large spatial queries using interior approximations. In *Symp. on Spatial and Temporal Databases*, pp. 404–421, 2001.

[9] S. Saltenis, C. Jensen, S. Leutenegger, and M. Lopez. Indexing the positions of continuously moving objects. In *ACM SIGMOD Int. Conf. on Management of Data*, pp. 331–342, 2000.

[10] Y. Theodoridis, T. Sellis, A. N. Papadopoulos, and Y. Manolopoulos. Specifications for efficient indexing in spatiotemporal databases. In *Conf. on Scientific and Statistical Database Management*, pp. 123–132, 1998.

[11] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *Int. Conf. on Scientific and Statistical Database Management*, pp. 111–122, 1998.