Extending Database Technology to Support Location-Based Service Applications

Dragan H. Stojanović[1] and Slobodanka J. Djordjević-Kajan[2]

Abstract – **Location-based service applications are based on mobile objects and management of their continuously changing locations. The goal of work presented in this paper is to provide extended database support for such applications, by defining mobile objects data model and an SQL extension, based on widely accepted OGC and ISO TC 211 specifications.**

Keywords – **Database, Location-Based Services, Mobile objects**

I. Introduction

Advances in wireless communication technologies, mobile positioning and Internet-enabled mobile devices, like smart phones and PDA, have given rise to a new class of location-based applications and services. Location-based services (LBS) deliver geographic information and geo-processing power to the mobile/static users in accordance with their current location and preferences, or location of the static/mobile objects of their interests. LBS are specialized, multi-tiered, component Web GIS applications, which can be published, located and invoked across the wired/wireless Web [7]. Such services like automatic vehicle location, fleet management, tourist services, transport management, traffic control and digital battlefield, are all based on mobile objects and management of their continuously changing locations. Thus, LBS applications require database and application support to model and manage mobile objects in both database and application domain and to support querying on the motion properties of the objects [8,9].

II. Mobile Point Objects in LBS

LBS are multi-tiered Internet GIS applications with architecture presented in Fig 1. The location of mobile information appliances can be determined by using GPS or mobile network triangulation. They can report their location to the LBS server through a wireless interface, or their location can be obtained through ground-based radars or satellites. At the LBS server, the data is processed and services, based on such data, are provided to the users. Mobile users may also represent mobile objects of interest to other users of the particular service. Mobile objects in LBS are characterized by point geometry and to the rest of the paper the term mobile object pertains to point object. According to T. Brinkhoff in [1] mobile real-world point objects always move along a path in a transportation network. Vehicles, trains, boats and passengers move following a particular network (roads, railways, rivers, pedestrian tracks). Air traffic also follows a (3-D) network of air corridors. Knowing their destinations, mobile objects often use fast and/or shortest paths depending on the cost criteria (time and/or distance).

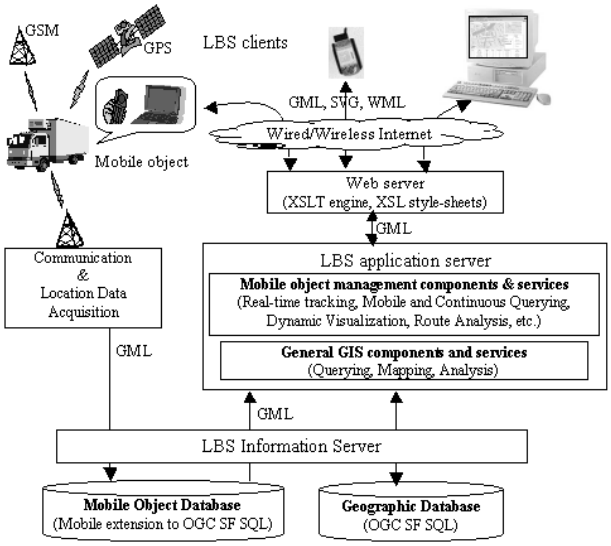

Fig. 1. The LBS architecture

The application scenario for LBS that involves mobile objects both as the users of the service and as tracked objects is as follows [8]. The mobile object registers for the certain location-based service connecting to the LBS server by sending the starting location (coordinates of the starting point or address), ending location, eventually set of points of interest that it is going to visit along its route and current speed (intended average speed). The LBS server retrieves to the mobile object its route and uncertainty threshold [9]. Based on the route and the current/average speed defined, an approximation of the expected motion of the object from the last registered location to the (near) future can be determined. The uncertainty threshold specifies the responsibility of the mobile object to send the location update to the LBS server if its current location is deviated from its expected location by defined uncertainty threshold. With every location update, the mobile object must also update the current/average speed to enable prediction of its future motion till the next location

[1]Dragan H. Stojanović is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia and Montenegro, Email: dragans@elfak.ni.ac.yu

[2]Slobodanka J. Djordjević-Kajan is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Nis, Serbia and Montenegro, Email: sdjordjevic@elfak.ni.ac.yu

update. Performing such scenario, mobile object generates its certain trajectory, as an approximation of its motion in space and time. The trajectory of the mobile object is a polyline in three-dimensional space (two-dimensional space and time) represented as a sequence of points (x_i, y_i, t_i). The reason it is only an approximation is that the object does not move in straight lines at constant speed. The number of points along the trajectory is proportional to the accuracy of such approximation. An additional parameter mt_i for every point defines the type of motion during period $[t_i, t_{i+1}]$. The four motion types are defined: *punctual* – the mobile object isn't tracked and its location isn't defined during certain time period; *stepwise* – the mobile object does not move during the time period; *linear* – the mobile object move along the straight line from (x_i, y_i) to (x_{i+1}, y_{i+1}), and at constant speed, and *interpolated* – the speed of the mobile object during defined time period isn't constant and must be defined by an interpolation function.

III. Modeling Mobile Objects in the Argonaut Framework

The standard-based component framework, named ARGONAUT, represents a suite of mobile object data modeling and management components [8]. The foundational data model of the ARGONAUT framework was developed to support conceptual modeling and querying of changing properties of geographic features. The ARGONAUT data model is extensible, object-oriented, and specified using UML class diagram notation. We base our modeling approach on the comprehensive framework of data types and rich algebra of operators defined in [3] but extending their approach to the object-oriented modeling paradigm, and representation of the current as well as the future motion of the mobile objects. Also, the main emphasis we put on the mobile point objects moving on the transportation network [1,8].

The ARGONAUT data model extends the OGC Simple Features model, also adopted by ISO TC 211 [4]. The standard defines abstract *Geometry* class, and its hierarchy of specialized geometric classes (*Point, LineString, Polygon, MultiPoint,* etc.). The attributes and operations defined within geometry classes support specification of topological relations, direction relations, metric operations and operations that support spatial analysis (point-set operations) on appropriate geometry types. Time dimension of a mobile object is specified through *TimeObject* class hierarchy, defined in accordance with ISO TC 211 Temporal Schema [5] (*TimeInstant, TimePeriod TimeDuration, MultiTimeInstant* etc.). The *TimeObject* class includes attributes and operations for specifying topological relations, metric operations and point set operations on time dimension.

The base class for introducing mobility of features and continuous change of their geometric properties is an abstract *MobileGeometry* class, as the root of the extensible hierarchy of classes for specifying mobile geometries (Fig. 2). For every class in Geometry class hierarchy an appropriate class for the representation of the mobile geometry is defined (*MobilePoint, MobileLineString, MobilePolygon, MobileMultiPoint,* etc.). Any of these classes appropriately restrict the Geometry class aggregated within the *MotionSlice* class (≪restriction≫ stereotype). Being a specialization of the *Geometry* class, a *MobileGeometry* and its specialized classes can be treated in the same way as any other geometric object, i.e. it can represent the geometric property of any feature which is dynamic in nature and participate in all geometric operations and relations. An instance of the *MotionSlice* class, aggregated by the *MobileGeometry* class with multiplicity 0..n, represents the registered location of a mobile geometry, by containing a geometry value (instance of *Geometry* class), the valid time of such value (instance of the *TimeInstant* class) and the motion type (value of enumeration type *MotionType*), which describes the way geometry changes between two successively registered geometries. The ARGONAUT data model provides four enumerated values for motion types, and those are: *Punctual, Stepwise, Linear* and *Interpolated*. The first three motion types don't require any additional information to be included in the data model, but for the *Interpolated* motion type, a reference to the *Interpolation* class, with defined interpolation parameters, must be specified. The motion of a mobile object also causes continuous change of its non-spatial properties like distance from some static or a mobile object or topological relation between two mobile objects, as elaborated in [2]. Using the same approach the ARGONAUT data model defines *MobileBoolean* and *MobileDouble* classes, by inheritance from *Boolean* and *Double* base classes respectively and aggregation of appropriate *MotionBoolSlices* or *MotionDoubleSlices* classes.

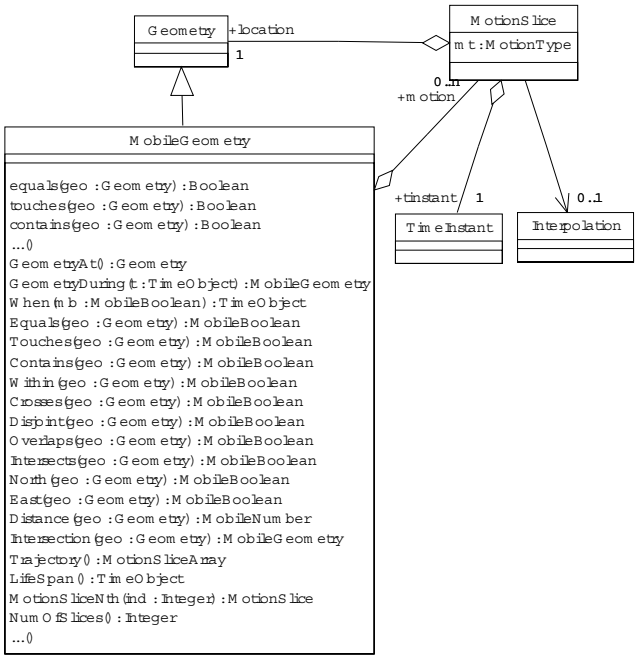

Fig. 2. The foundation of the ARGONAUT data model

The ARGONAUT data model overrides topological relations, direction relations, metric operations and spatial analysis operations inherited from the base *Geometry* class [4].

These relations and operations take mobile or non-mobile geometry argument and generate non-mobile results (Boolean, Double, Geometry, etc.). Specialized topological and direction relations, like *touches* and *contains*, have single argument of type *Geometry* (which could also be *MobileGeometry*) and return Boolean true value indicating that such relations are satisfied during the lifespan of the *MobileGeometry* argument(s) according to temporal aggregation defined in [2]. Such operations correspond to spatio-temporal predicates. Metric and spatial analysis overridden operations can not be considered as predicates, and for *MobileGeometry* argument operation is delegated by default to the geometry value at the current time instant defined using *GeometryAt(Now)* operation. The ARGONAUT data model also defines mobile variants of mentioned non-mobile relations and operations, named with starting capital letter in Fig. 2. Such operations correspond to the temporally lifted operations that handle non-mobile or mobile geometry arguments and generate mobile result, since for mobile argument(s) different results are generated in the course of time. The motion type associated with the mobile result depends on the operator or relation. In general, it is not sufficient to inherit a motion type of a mobile argument. For example, for mobile points characterized by linear motion type, a quadratic interpolation method is needed to represent the mobile result of distance operation.

To support the querying of mobile objects following operations are defined. The *Lifespan* operation returns time object defining the whole life span of the mobile object. The aforementioned *GeometryAt* operation returns geometry value at specific time instant. In order to restrict the sequence of motion slices of a mobile object according to specific time object, *GeometryDuring* operation is defined. That operation restricts the mobile object to only those motion slices from the sequence that belong to the specified time object given as an argument. The *When* operation returns the time object during which the mobile object satisfies the criteria specified by a mobile Boolean argument. The *Route* operation returns *Geometry* result representing the path traversed by the mobile geometry. All these operations are overridden in specific mobile classes inherited from the *MobileGeometry* class. To support manipulation of mobile properties of type *MobileBoolean* and *MobileDouble*, predicates and operations (*and, not, max, add*, etc) and their temporally lifted counterparts (*And, Not, Max, Add*, etc.), are defined accordingly [8].

Modeling mobile point objects that move continuously over a predefined network infrastructure, as existed in LBS, are provided by the *MobilePoint* class (Fig. 3). To determine the current location of a mobile point object, based on the last location update, the attribute *speed* is defined. If a mobile object does not follow the predefined route, the attribute *direction* must be defined accordingly. To specify a predefined route of the mobile point, the *MobilePoint* class is associated to the *Route* class which aggregates ordered set of *RoadSegment* class objects with OGC SF *LineString* geometry, and contains starting and destination points that lays on the first/last road segment respectively (Fig. 3). The *Route* class defines operations for specifying point along the route on specific distance from defined point (*PointOnDistance*) and the route distance between two points on the route (*RouteDistance*). The *GetRoute* method enables retrieval of *LineString* object, which represents the route geometry, while the *RoadSeg* operation retrieves the road segment along the route, which contains the specified point. The *MobilePoint* class inherits the *MobileGeometry* class and thus inherits and overrides aforementioned spatio-temporal operations and relations. The *Route* operation defined in *MobileGeometry* class is redefined in inherited classes, because for different mobile geometry types, the trajectory operation returns specific geometric result. Such operation applied to mobile line string object yields *Polygon* object as result, and for mobile point object with associated linear motion function returns a *LineString* geometric object.

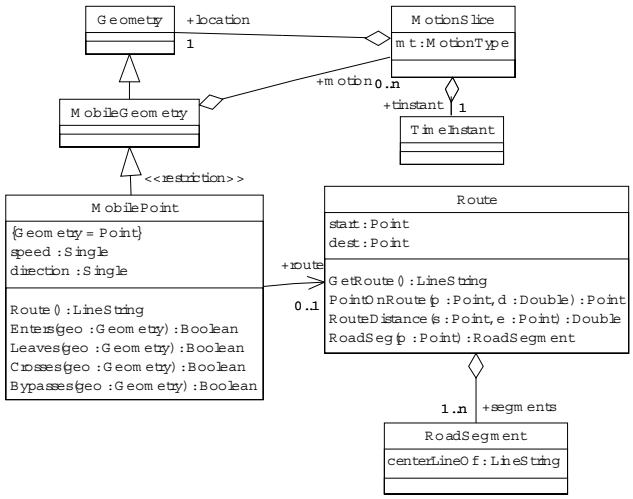

Fig. 3. Modeling mobile points on the transport network

The model defines relations arisen from the motion of mobile point over mobile or static polygonal area, such as *Enters, Leaves, Crosses, Bypasses*, etc. as proposed in [2] (Fig. 3). We define those relations for a mobile/static point and a polyline using the same semantics and definitions given in [2] for a mobile point and a region. Those operations are particularly useful in querying mobile points moving on network paths. Such relations can be defined by successive application of several basic mobile relations and enable examination of changing spatial relations over time by simply sequencing basic spatio-temporal predicates. Thus mobile object enters in the area of static or mobile polygonal object during given time period, if it was outside of the polygon at the beginning of the period (Disjoint relation), then at certain time instant was at the border of the polygon (touches relation) and is within the region to the rest of the time period (Within relation). Similar definitions hold for *Leaves* (inverse of *Enters*), *Crosses* and *Bypasses* predicates.

IV. Implementation in an Object-Relational DBMS

The implementation of proposed mobile object data model in an object-oriented application and a database is straightforward using defined UML class diagrams. The data model implementation in (object-) relational database domain is based

on definition of user-defined data types and operations within Object-Relational DBMS and appropriate mobile extension of OGC Simple Features for SQL specification [4]. If the LBS server is based on a relational DBMS, support for mobile object data management is integrated in LBS application components. User-defined data types and user-defined functions are defined in terms of the SQL DDL statement CREATE TYPE [6] as follows:

```
CREATE TYPE MotionSlice AS OBJECT
(slice#  NUMBER, geometry Point, timeinst DATE, mt MotionType);

CREATE TYPE Motion AS TABLE MotionSlice;

CREATE TYPE MobilePoint AS OBJECT
(GID  NUMBER, speed NUMBER, direction NUMBER, motion
Motion,
MEMBER FUNCTION GeometryAt(tTimeInstant) RETURN Point
… );

CREATE TYPE Ambulance AS OBJECT
(ID NUMBER, name CHAR(64), driver CHAR(64), type CHAR(256),
locations MobilePoint);

CREATE TYPE Taxicab AS OBJECT
(ID NUMBER, company CHAR(64), driver CHAR(64), type
CHAR(256), locations MobilePoint);

CREATE TYPE Street AS OBJECT
(ID NUMBER, name CHAR(64), centerLineOfLineString);
```

The operations defined for those types are implemented as user defined functions that can be applied to user defined data types. The query facility of SQL is provided by the well-known SELECT-FROM-WHERE clause. User defined operations on user defined data types can be also included as predicates and functions within SELECT and WHERE clause and embedded in an SQL statement. Thus, mobile spatio-temporal queries can be specified and processed, like:

1. Select ambulances that are within 2 km around of my current address:

```
select amb.id, amb.name, amb.driver
from Ambulance amb
where
(amb.locations.GeometryAt(NOW)).Distance(Point(xref,yref))  <=
2000
```

We assume that the LBS application provide geocoding possibilities that convert location expressed as address in point value with exact coordinates xref and yref. Thus *GeometryAt* operation returns the location of a mobile point at the specified time instant (NOW defines the current time). Operation *Distance* returns the distance between point values and the whole predicate in the WHERE clause specifies only those ambulances whose determined distance is less then 2000 meters.

2. Return the length of path of truck "BioExport-1" in kilometers and time:

```
select (t.troute.Trajectory()).length(), (t.troute.Lifespan()).duration()
from Truck t
where t.name="BioExport-1"
```

Trajectory operation returns *LineString* object and the *length* operation calculates length of that line string. Similarly, *Lifespan* operation returns *TimeObject* object, and *duration* operation applied on *TimeObject* argument returns its duration.

3. Return the position of a taxicab "Banker-17" if it enters the "Cara Dusana" street in last 5 minutes:

```
select t.locations.GeometryAt(NOW)
from Taxicab t, Streets
where t.name = "Banker-17" and s.name="Cara Dusana" and
    (t.locations.GeometryDuring(TimePeriod(NOW -
300,NOW))).Enters(s.centerLineOf)
```

Operation *Enters* is applied to the *MobilePoint* object whose motion is restricted by time period value which correspond to the "last 5 minutes" expression, and the *LineString* object representing the center line property of the street object.

V. Conclusion

The main contribution of this paper is the object-oriented data model and SQL extension for representation and querying mobile objects. It represents the foundation of the ARGONAUT component framework for development of location-based services that involve mobile objects. Such framework can be easily and effectively integrated with any OGC-compliant object-relational DBMS to provide mobile object data management and querying capabilities. We are currently developing a prototype application for tourist and business guiding based on proposed data model and the ARGONAUT component framework implemented over Web architecture.

Acknowledgement

The research was partially supported by the project "Geographic Information System for Local Authorities based on Internet/WWW Technologies", funded by Ministry of Science, Technology and Development, Republic of Serbia, and Municipality of Nis, Contract No. IT.1.23.0249A.

References

[1] T. Brinkhoff, "A Framework for Generating Network-Based Moving Objects", *GeoInformatica Journal*, Vol. 6, No. 2, June 2002, pp. 153-180.

[2] M. Erwig and M. Schneider, "Spatio-Temporal Predicates", *IEEE Trans. On Knowledge and Data Engineering*, Vol 14, No. 4, July/Aug. 2002, pp. 881-901.

[3] R. H. Gueting, M. H. Boehlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider and M. Vazirgiannis, "A Foundation for Representing and Querying Moving Objects", *ACM-Transactions on Database Systems Journal*, Vol. 25, No. 1, 2000, pp. 1-42.

[4] ISO/ TC 211 Geographic Information/Geomatics, ISO 19125 - Geographic information - Simple feature access. Oct. 2000.

[5] ISO/ TC 211 Geographic Information/Geomatics, ISO 19108 - Temporal Schema. Oct 2000.

[6] Oracle 9i Enterprise Edition, Oracle Documentation Library, Oracle Corporation, http://technet.oracle.com, 2002.

[7] D. Stojanović and S. Djordjević-Kajan, "Location-based Web service for tracking and visual route analysis of mobile objects", *Proc. of Yu INFO 2002*, Kopaonik, 2002.

[8] D. Stojanović and S. Djordjević-Kajan, "Modeling Mobile Point Objects in Location-based Services", *Int. Journal Facta Universitatis: Mathematic & Informatics*, accepted for publication, 2003.

[9] O. Wolfson, "Moving Objects Information Management: The Database Challenge", 5th Workshop on Next Generation Information Technologies and Systems (NGITS 2002) Israel, 2002.