A Spatial-Temporal Object Database Approach to Dynamic Segmentation

Bo Huang#
Assistant Professor
Department of Civil Engineering
National University of Singapore
1 Engineering Drive 2, E1A #08-06
Singapore 117576
Telephone: +65 6874 2158
Fax: +65 779-1635
Email: cvehb@nus.edu.sg

Li Yao
Research Scholar
Department of Civil Engineering
National University of Singapore
1 Engineering Drive 2, E1A #08-25
Singapore 117576
Telephone: +65 6874 5035
Email: engp2469@nus.edu.sg

Submitted in Response to Committee A5015’s CFP
for Presentation and Publication at the
82nd Annual Meeting of the Transportation Research Board
January 12-16, 2003

Word count: 6699
ABSTRACT

Dynamic segmentation is viewed as one of the most important functions of GIS for transportation applications. Although the road network and associated events (e.g. pavement material, traffic volume and incidents) can be referenced to both space and time, the spatial and temporal dimensions has not been well integrated. This paper explores how to model space-varying, time-varying and space-time-varying events in dynamic segmentation using an object database approach that is in line with the Object Database Management Group (ODMG) standard. A mechanism, called parametric polymorphism, is utilized to lift conventional data types to spatial, temporal and spatial-temporal types for maintaining knowledge about events that could change spatially, temporally, and spatial-temporally along linear features. An associated object query language, called DS-OQL, has also been designed to support the formulation of spatial, temporal and spatial-temporal queries on the road and event information.

Keywords: GIS-T; Dynamic Segmentation; Object Model; Object Query Language (OQL); Parametric Polymorphism; Parameterized Types
INTRODUCTION

GIS-T, a synergy of GIS and transportation, has been receiving increased attention in recent years (6, 7, 9, 15, and 19). Currently, a variety of GIS-T applications covering many areas in transportation exist, such as travel demand analysis, land use and transportation interaction, and route planning. Common to these applications is that they are often tied to transportation networks. The modeling of these networks is, however, a non-trivial task as the networks have complex properties (e.g., links, nodes, turns, and modal transfers). Further, there are events (e.g., speed limit, pavement material, accidents and bus stops) associated with them, which may change over both space and time. Of the various GIS-T data models used for linear referencing, dynamic segmentation, a powerful tool for representing linear attributes and events, is commonly viewed as an aspect of GIS that provides an effective means when a transportation agency is deploying a GIS.

Dynamic segmentation is a process of dynamically locating events on linear features—that is, directly from attribute tables of events for which distance measures are available (8, 16). Thus, it provides the ability to integrate various event information in linear measure (e.g., milepost) into base networks. This, in turn, allows multiple sets of attributes to be associated with any portion of a linear feature without affecting the underlying linear features’ x, y coordinates. Dynamic segmentation has been implemented in several commercial GIS packages such as Arc/Info, ArcView, TransCAD and GeoMedia based upon relational data models. As such, events are commonly stored in separate relational tables; each table can be linked to a route and queried. When a query involves more than one event, the tables have to be specially joined, with each record in the joined table representing a new section. As the entire representation of events is forcibly decomposed into a complex set of tables, it is unnatural and results in low efficiency in query execution. Also, the integrity of the whole image is segmented.

In contrast to relational data models, an object-oriented (OO) model allows a more natural representation of real-world concepts in the database (21). Techniques for improving the methods for GIS data representation using an OO approach have been explored (21). OO GIS have also been designed and prototyped (10, 11). However, such endeavors are rare in the area of GIS-T (13). An important reason is probably the lack of a common framework for complex transportation data modeling that is crucial for further applications. But this is not true any more since some generic conceptual data models have been developed such as the GIS-T enterprise data model (6), the generic model for linear referencing (20), and the multidimensional location referencing system (MDLRS) data model (1). These models lay a basis for the further work towards object-oriented data modeling that is the focus of this study. Previously, the lack of an agreement for a common OODB data model and query language has hindered the widespread diffusion of OO Databases (OODBs) (12). However, a consensus is now emerging, with the Object Data Management Group (ODMG) effort on standardizing OODBs and the recent release of ODMG 3.0 (5). The proposed ODMG standard comprises an object model (ODMG Object Model), an object definition language (ODL) and an object query language (OQL). This study concerns an extended ODMG object model and query language for dynamic segmentation.
OODBs were developed to deal with complex database applications (2). Thus, they are equipped with powerful data abstractions and modeling facilities. Because dynamic segmentation involves multiple sets of attributes which vary spatially and temporally, and require complex type support and advanced modeling concepts, OODBs are excellent tools for realizing such a representation without requiring fundamental extensions to the basic data model (12, 14). The challenge is, however, how to augment an object data model to capture the spatial and temporal changes of attributes and to augment an object query language to permit queries over such changes. As different events can have different distributions on a route, each of them that might be of a different type (e.g. speed limit is of type Integer and pavement quality is of type String) requires maintaining changes over a linear feature. To accomplish this, the ODMG object model is extended with parametric polymorphism (or generics) which uses types as parameters in generic type (or class) definition.

Parametric polymorphism has been extensively studied in programming languages (PL) especially the functional PLs (e.g. Metalanguage, ML). This notion has also been introduced to OODBs (3). Significantly, this form of polymorphism offers a mechanism, which allows a function to work uniformly on a range of types; these types normally exhibit some common structure (4). Consequently, three parametric classes, Spatial\(<T>\), Temporal\(<T>\) and ST\(<T>\), are defined, which lift type \(T\) into a spatial type that contains the distribution of all sections of \(T\) (e.g. all sections of pavement material of type String), a temporal type that contains the history of all episodes of \(T\), and a spatial-temporal type that contains both the distribution and history of a road segment or an event, respectively. The parameter type \(T\) can be any ODMG type such as Integer, String and Struct. Corresponding operations inside the parameterized types are also provided to traverse the distribution of attributes. This polymorphism allows users to define the type of any attribute as spatially varying, temporally varying or spatial-temporally varying, thereby supporting the modeling of events. The three parameterized types defined in this paper represent a useful extension to ODMG to meet the requirement of modeling and queries with respect to dynamic segmentation.

This paper seeks to provide a packaged solution to dynamic segmentation using a new object database approach. An object model has been designed, which extends the ODMG object model with parametric polymorphism to support the definition of parameterized types, facilitating representation of events that may change in both space and time along a route. In addition to this model, we design a query language by extending OQL with a number of syntactic constructs to manipulate spatially and temporally varying information associated with the events.

The remainder of this paper is organized as follows. The next section describes the principle of dynamic segmentation in conjunction with linear referencing, and analyzes the problems of current dynamic segmentation models. The object model developed in this study is then presented with emphasis on three parameterized types. This is followed by an associated query language for analysis, together with some demonstrative examples. Finally, this paper is concluded with a summary and outlook.
EVENTS AND SPATIO-TEMPORAL DYNAMIC SEGMENTATION

Linear referencing is a useful tool for transportation because many transportation features are linear in nature. In linear referencing, location is given in terms of a known feature and a position or measure on it. For example, route I-10, kilometer 23, uniquely identifies a position in geographic space without having to express it in (x, y) coordinates or (latitude, longitude) terms.

Dynamic segmentation is generally a two-step process performed on linear features. First, a route system is created from a base network that can represent, for example, transit bus routes, shortest path from point A to B, and so forth. Each route in a route system is a connection of links or partial links. Second, events are associated with the route system by using measures. While there are different linear referencing methods (e.g. 20), a GIS often takes the distance from the starting point of each route as an event measure. This approach is also adopted in this study. The classes of interest in the dynamic segmentation model are shown in Figure 1.

Dynamic segmentation allows multiple sets of attributes to be associated with any portion of a linear feature. These attributes can be stored, displayed, queried, and analyzed without affecting the underlying linear data's (x, y) coordinates. Dynamic segmentation models linear features using routes and route events. A route represents a linear feature such as a city street, highway, or river. Routes contain measures, which describe the distance along them. The measures are used to locate data, which describes parts of the route. Data along routes is modeled using route events. An event may change over both space and time (19). For example, a section of a route may be extended in length. This is change in location. At the same time, its pavement quality may change as time goes on.

Dynamic segmentation is, in fact, conducted in terms of events on a route. By spatial segmentation, we mean the segmentation in terms of event locations and by temporal segmentation, we mean the segmentation in terms of time. Naturally, spatio-temporal segmentation refers to the segmentation of a route in terms of both location and time. These segmentation approaches are reflected by queries involving both location and time constraints on the events. Therefore, an object model is devised below to represent the events that change over space, time or both of them. An associated query language is also developed to perform queries on the events in terms of location and time.

OBJECT MODEL

Extended Data Types

ODMG provides neither spatial types nor temporal types except Temporal Structured literals (e.g., Date, Time and Timestamp). The objective of this paper is to extend the ODMG Object Model with spatial, temporal, and spatial-temporal parameterized types and to illustrate the interest of these extensions at the query language level. The extended types in this paper are shown in Figure 2.

The ODMG standard provides an object definition language, ODL, for defining the object types that conform to the ODMG Object Model. DS-ODL extends ODL with a
number of class definitions and syntactical constructs that accommodate spatial-temporal information. DS-ODL is used in class definitions throughout this paper.

**Basic Spatial and Temporal Types**

**Spatial Types**

The geometric types: **Geometry**, **Point**, **LineString**, **Polygon**, **GeometryCollection**, **Points**, **LineStrings** and **Polygons** are utilized for specifying a spatial feature. **Geometry** is the root class of all these types, and is an abstract (non-intangible) one. It includes operations that access properties of objects (e.g. envelope), determine topological relationship (e.g. overlaps, meets, disjoint, etc.), and support spatial queries (e.g. distance, buffer, intersection, union, etc.).

```plaintext
Class Geometry {
    //dimension 0-Point, Points
    // 1-LineString, LineStrings
    //2-Polygon, Polygons
    attribute Unsigned Short dimension;

    //accessing spatial properties
    Struct MBR {Double minx; Double miny; Double maxx; Double maxy;};
    MBR envelope(in Geometry g);

    //test topological relationship
    Boolean disjoint (in Geometry g);
    Boolean intersects (in Geometry g);
    Boolean crosses (in Geometry g);
    Boolean contains (in Geometry g);
    Boolean overlaps (in Geometry g);
    ...

    //create new spatial objects
    ...
    Geometry intersection (in Geometry g);
    Geometry difference (in Geometry g);
}
```

There are also other methods defined in the sub-classes like length for the **LineString** type and area for the **Polygon** type. For more details about these types and operations refer to (17).

**Space Interval**

Space interval represents a section with the same property (e.g. good pavement quality) on a route. It is defined as follows:
class Space_Interval
{
    Attribute Double SI_from;
    Attribute Double SI_to;

    Unsigned Long length();

    //possible spatial relationship
    Space_Interval intersection (in Space_Interval loc);
    Space_Interval union (in Space_Interval loc);
    Space_Interval none (in Space_Interval loc);

    //test spatial relationship
    Boolean equals (in Space_Interval loc);
    Boolean contains (in Space_Interval loc);
    Boolean meets (in Space_Interval loc);
    Boolean overlaps (in Space_Interval loc);
    Boolean in_front_of (in Space_Interval loc);
    Boolean starts (in Space_Interval loc);
    Boolean ends (in Space_Interval loc);
}

This class has two attributes, SI_from and SI_to, both specified by a measure of type Double, which indicates the beginning and the end of a space interval. A space interval can be constructed as shown in Table 1.

The space interval, [location], contains just one event point, whiles the space interval, [location1, location2], contains all the points on the route between and including location1 and location2. Therefore, [location] is used to locate a point event, while [location1, location2] is used to locate a linear event. Operations for comparing two space intervals a and b, such as a in_front_of b, are provided in the Space_Interval class. The intersection and union of space intervals are also provided. Note that a section representing a space interval can be in any linear shape, not necessarily a straight line.

Time Interval

Time interval represents a period of time. Its class definition is as follows:

class Time_Interval
{
    Attribute Timestamp TI_start;
    Attribute Timestamp TI_end;

    Unsigned Long duration ();

    //possible temporal relationship
TimeInterval intersection (in TimeInterval tv);
TimeInterval union (in TimeInterval tv);
TimeInterval none (in TimeInterval tv);

//test temporal relationship
Boolean equals (in TimeInterval tv);
Boolean contains (in TimeInterval tv);
Boolean meets (in TimeInterval tv);
Boolean overlaps (in TimeInterval tv);
Boolean before (in TimeInterval tv);
Boolean starts (in TimeInterval tv);
Boolean finishes (in TimeInterval tv);
}

The class TimeInterval has two attributes start and end, both of ODMG type Timestamp, which indicate the beginning and the end of a time interval. TI can be constructed as [TI_start, TI_end], which represents all the time points between and including TI_start and TI_end. If TI_start equals TI_end, TI contains just one time point, simply represented as [TI_start] or [TI_end]. This class defines a number of elementary temporal operations for comparing two time intervals such as equals, contains and before, as well as those for the intersection and union of two time intervals. A time interval can be constructed as shown in Table 2.

Parameterized Types

Parameterized types come with parametric polymorphism. Polymorphism means the ability to take several forms. In object-oriented programming, it refers to the ability of an entity to refer at run-time to instances of various classes; the ability to call a variety of functions using exactly the same interface (as is provided by virtual function).

Parametric polymorphism is a new feature representing an improvement on the original polymorphic features of the language. It provides a way of reducing the amount of code, and at the same time of making functions and types really generic, independent on the argument types (for functions) and contained types (for types).

In general, parametric polymorphism provides the possibility of writing classes that operate on data without specifying the data’s type. In other words, a generic type can be formulated by lifting any existing type. A simple parametric class has the following form:

```java
class CP<parameter>{
    parameter a;
    ...
};
```

where parameter is a type variable. The type variable can be of any ODMG type, which may be used inside the declaration of CP wherever a type name is permitted.
The Spatial\(<T>\) Type

As the values of events (e.g. speed, pavement quality and accident) change along a linear feature (i.e. route), a parameterized type called **Spatial\(<T>\)** is defined. Each type \(T\) is defined to reflect the change of events over space:

```scala
class Spatial <T> {
    struct (T val, Space -interval loc) section;
    attribute List <Section> distribution;

    //Projection operations
    List <Section> ProjectionByLoc (in Space_Interval loc);
    List <Section> ProjectionByVal (in T val);

    //operations for obtaining an index number of a section
    Unsigned Long GetIndexByVal (in T val);
    Unsigned Long GetIndexByLoc (in Line_Interval loc);
}
```

The **distribution** of an attribute of type \(T\) is represented by a list of pairs of type **section**. In this sense, the value of a linear attribute (i.e. an event) of type **Spatial\(<T>\)** is expressed as a linearly ordered (from the beginning of a route to its end) set of sections (i.e. value-location pairs):

\[
\{(\text{val}_1, \text{loc}_1), (\text{val}_2, \text{loc}_2), \ldots, (\text{val}_n, \text{loc}_n)\}
\]

where \(\text{val}_1, \ldots, \text{val}_n\) are legal values of type \(T\), \(\text{loc}_1, \ldots, \text{loc}_n\) are line intervals on condition that \(\text{loc}_i \cap \text{loc}_j = \emptyset, i \neq j\) and \(1 \leq i, j \leq n\). Each pair, i.e., \((\text{val}_i, \text{loc}_i)\), represents a section of the structure type section. As a result, any ODMG type \(T\) can be promoted to a linear type representing a collection of \(<T, Space\_Interval>\) values. In other words, a **Spatial\(<T>\)** type adds a spatial dimension (represented in measures) to \(T\). This is crucial for dynamic segmentation modeling because the spatial changes of an event along a linear feature can be maintained.

The Temporal\(<T>\) Type

Just as **Spatial\(<T>\)** which can maintain the change of events over space, **Temporal\(<T>\)** is able to represent the change of events over time, whose schema is shown below:

```scala
class Temporal<T> {
    struct (T val, Time_Interval tv) State;
    attribute List <State> history;

    //projection operations
```
A **Temporal** object is a temporally ordered collection of **value-timeinterval** pairs:

\[ \{(v_{1}, t_{v1}), (v_{2}, t_{v2}), \ldots, (v_{n}, t_{vn})\} \]

where \(v_{1}, \ldots, v_{n}\) are legal values of type \(T\), \(t_{v1}, \ldots t_{vn}\) are time intervals such that \(t_{v_i} \cap t_{v_j} = \emptyset, i \neq j\) and \(1 \leq i, j \leq n\). The parameter type \(T\) can be any ODMG type, and so an ODMG type is lifted into a temporal type.

**The ST\(<T>** Type

\(ST<T>\) represents the change of events over both space and time.

```cpp
class ST<T>
{
    struct (T value, SpaceInterval loc; TimeInterval tm) section_state;

    attribute List <Section> distribution_history;

    //Projection operations
    List <Section> ProjectionByVal (in T val);
    List <Section> ProjectionByLoc (in Space_Interval loc)
    List <Section> ProjectionByTv (in Time_Interval tv);

    //operations for obtaining an index number of a section
    Unsigned Long GetIndexByVal (in T val);
    Unsigned Long GetIndexByLoc (in Space_Interval loc);
    Unsigned Long GetIndexByTv (in Time_Interval Tv);
}
```

A ST object is a collection of **value-location-timeinterval** triplets:

\[ \{(v_{1}, l_{oc1}, t_{m1}), (v_{2}, l_{oc2}, t_{m2}), \ldots, (v_{n}, l_{ocn}, t_{mn})\} \]

where \(v_{1}, \ldots, v_{n}\) are legal values of type \(T\), \(l_{oc1}, \ldots l_{ocn}\) are line intervals on condition that \(l_{oc_i} \cap l_{oc_j} = \emptyset, i \neq j\) and \(1 \leq i, j \leq n\), \(t_{m1}, \ldots t_{mn}\) are time intervals such that \(t_{m_i} \cap t_{m_j} = \emptyset, i \neq j\) and \(1 \leq i, j \leq n\). Each pair, i.e., \((v_{i}, l_{oc_i}, t_{m_i})\), represents a section state of the structure type **section_state**. As a result, any ODMG type \(T\) can be promoted to a collection of \(<T, Space_Interval, Time_Interval>\) values.
An Example Utilizing Parameterized Types

Using the Spatial<T>, Temporal <T> and ST <T> types, a route class is defined which fits well into the dynamic segmentation framework.

```java
class route
    (extent routes key ID)
{
    attribute String ID;
    attribute String category;
    attribute Spatial<String> pvmt_quality; //user-defined space-varying event
    attribute Spatial<Integer> max_speed;    //user-defined space-varying event
    attribute Temporal<Integer> lane_closure; //user-defined time-varying event
    attribute ST<Integer> accident;          //user-defined space-time-varying event
    attribute Linestring shape;
};
```

The types of events pvmt_quality and max_speed are lifted to Spatial<String> and Spatial<Integer>. These attributes are now capable of representing the distribution of sections along a route by associating locations with their value changes. The attribute lane_closure is lifted to Temporal<Integer>, which indicates some lanes may be closed at some time. The attribute accident is lifted to ST<Integer>, which associates both location and time to a traffic accident. However, the other three attributes (i.e. ID, category, and shape) remain intact. Therefore the Spatial<T> type, Temporal<T> type and ST<T> types allow users to choose which attributes to lift or not.

QUERY LANGUAGE

Basic Principle

A query language provides a high-level interface for users to interact with the data stored within a database. Although some queries in relation to dynamic segmentation can be entertained by some GIS packages, a formal interface similar to OQL, which is able to support the retrieval of event and route information, does not exist. DS-OQL, an object query language for spatio-temporal dynamic segmentation, has therefore been constructed on top of DS-OM.

To date, there is no consensus for the taxonomy of spatio-temporal queries. However, some common requirements have been identified (12) such as temporal selection, spatial selection and spatio-temporal join.

DS-OQL extends OQL facilities to retrieve spatial, temporal and spatial-temporal information. The sections in a distribution, states in a history and section-states in a
distribution history are extracted through iteration in the OQL from-clause. Constraints in the where-clause can then be applied to the value, timestamp or location of a section through corresponding operations that have been defined in the above model part. Finally the result is obtained through the projection operation in the select-clause.

To illustrate the expressions of DS-OQL, a road network in Singapore is employed. CTE (Central Expressway) is selected as a route for our study. The geometry of the expressway varies from two to four lanes in each direction of travel. On the average, there are three lanes and a road shoulder in each direction over most of its length. The CTE serves a wide geographical area and provides vehicular access to major housing estates, business areas, and other major expressways in Singapore. The average daily traffic flow along different sections of the CTE typically varies from 454000 to 1440000 vehicles per day. Figure 3 provides an overview of the Singapore road network and the location of CTE, while Figure 4 gives an example of the event representation.

**Syntactical Constructs**

Given the above principle, DS-OQL provides some syntactical extensions to OQL to manipulate space-varying, time-varying and space-time-varying information represented by \texttt{Spatial<T>}, \texttt{Temporal<T>} and \texttt{ST<T>}, respectively.

In Table 3, \texttt{time1} and \texttt{time2} are expressions of type \texttt{Timestamp}, \texttt{location1} and \texttt{location2} are expressions of type \texttt{Double}. \texttt{e} is an expression of type \texttt{Temporal<T>}, and \texttt{es} is an expression denoting a state within a history, a section within a distribution, or a section state within a distribution history. Table 3 shows the different syntactical constructs in DS-OQL.

**Query Examples**

The following examples illustrate different types of spatial-temporal queries on the events in the route class.

**Example 1 (spatial selection).** Find the speed limit from 2 to 4 miles.

```sql
select r_speed.val
from routes as route, route.pvmt_quality! as r_qlty,
     route.max_speed! as r_speed
where r_qlty.loc.overlaps([2,4])
```

In this query, variable \texttt{r_qlty}, ranging over the distribution of route.pvmt_quality, represents a pavement quality section. \texttt{r_qlty.loc} returns the location of a pavement quality section. The \texttt{overlaps} operator in the where-clause specifies the condition on a pavement quality section’s location.

**Example 2 (spatial projection).** Display the location of accidents on route 1 where the maximum speed is 60 mph and pavement quality is poor.

```sql
select r_acdt.loc
```
from routes as route, route.acdt! as r_acdt,  
    route.pvmt_quality! as r_qly, route.max_speed! as r_speed  
where route.ID="1" and r_speed.val = 60 and r_qly.val = "poor" and  
    (r_speed.loc.intersection(r_qly.loc)).contains(r_acdt.loc)  

The intersection operation in the where-clause obtains the common part of r_speed.loc  
and r_qly.loc.

Example 3 (temporal projection). Show the time of all accidents on Route 1.

select r_acdt.tm  
from routes as route, route.acdt! as r_acdt  
where route.ID="1"

In this query, variable r_acdt, ranging over the distribution of route.acdt, represents an  
accident. r_acdt.tm returns the time of a selected accident.

Example 4 (spatial and temporal join). Find the accidents which occurred at 8:00-  
8:30am on route 1 where the pavement quality is poor.

select r_acdt.loc  
from routes as route, route.pvmt_quality! as r_qly,  
    route.accident! as r_acdt  
where Route.ID="1" and r_acdt.tm.overlaps([8am, 8:30am]) and r_qly.val = "poor" and  
    r_qly.loc.contains(r_acdt.loc)  

This query join two events through the contains operator.

CONCLUSIONS

This paper has explored a generic approach to adding multiple dimensions to  
conventional data types to represent spatially and temporally varying information. Then a  
route can be segmented by applying spatial and temporal constraints to the events. A neat  
spatio-temporal object model to dynamic segmentation has been developed by extending  
the ODMG OM with parametric polymorphism. This mechanism allows any ODMG type  
to be enhanced into a collection type that assigns a distribution, history or both to any  
typed event. Compared with the representation of dynamic segmentation in a relational  
database setting, the object model in this paper provides a unified view of routes and  
events in a natural and integrated way. An associated object query language has also been  
provided to manipulate space-time varying information. Its extensions to OQL are  
minimal, uniform, and easy to understand.

The model and query language are currently under an active implementation. While  
the object model developed in this paper represents the events that change over both  
spatial and temporal constraints to the events. A neat spatio-temporal object model to  
spatially and temporally varying information. Then a route can be segmented by  
applying spatial and temporal constraints to the events. A neat spatio-temporal object  
model to dynamic segmentation has been developed by extending the ODMG OM with  
parametric polymorphism. This mechanism allows any ODMG type to be enhanced into  
a collection type that assigns a distribution, history or both to any typed event. Compared  
with the representation of dynamic segmentation in a relational database setting, the  
object model in this paper provides a unified view of routes and events in a natural and  
integrated way. An associated object query language has also been provided to manipulate  
space-time varying information. Its extensions to OQL are minimal, uniform, and easy  
to understand.

The model and query language are currently under an active implementation. While  
the object model developed in this paper represents the events that change over both  
space and time naturally and elegantly, like the models in other object-oriented systems,  
it does not necessarily indicate that it can also improve the segmentation efficiency. We  
have implemented a preliminary interface between the object model and the object-
relational model in Oracle/Spatial. Through this interface, the spatio-temporal segmentations are conducted through queries.

REFERENCES

LIST OF TABLES

TABLE 1 Construction of Space Intervals
TABLE 2 Construction of Time Intervals
TABLE 3 Syntactic Constructs in DS-OQL

LIST OF FIGURES

FIGURE 1 An entity-relationship diagram for dynamic segmentation (adapted from (6))
FIGURE 2 Extended object types
FIGURE 3 Overview of the Singapore road network
FIGURE 4 An example representation of events on a route
TABLE1 Construction of Space Intervals

<table>
<thead>
<tr>
<th>Operation</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[measure]</td>
<td>Space_Interval</td>
</tr>
<tr>
<td>[measure1, measure2]</td>
<td>Space_Interval</td>
</tr>
</tbody>
</table>
TABLE2 Construction of Time Intervals

<table>
<thead>
<tr>
<th>Operation</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[time]</td>
<td>Time_Interval</td>
</tr>
<tr>
<td>[time1, time2]</td>
<td>Time_Interval</td>
</tr>
</tbody>
</table>
### TABLE 3 Syntactical Constructs in DS-OQL

<table>
<thead>
<tr>
<th>DS-OQL</th>
<th>Spatial&lt;T&gt;</th>
<th>Temporal&lt;T&gt;</th>
<th>ST&lt;T&gt;</th>
<th>Result Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[location1, location2]</td>
<td>struct(start:location1, end:location2)</td>
<td>struct(start:location1, end:location2)</td>
<td></td>
<td>SpaceInterval</td>
</tr>
<tr>
<td>[time1, time2]</td>
<td>struct(start:time1, end:time2)</td>
<td>struct(start:time1, end:time2)</td>
<td></td>
<td>TimeInterval</td>
</tr>
<tr>
<td>E!</td>
<td>e.distribution</td>
<td>e.history</td>
<td>e.distribution_history</td>
<td>List</td>
</tr>
<tr>
<td>es.val</td>
<td>es.val</td>
<td>es.val</td>
<td>es.val</td>
<td>T (any ODMG type)</td>
</tr>
<tr>
<td>es.vt</td>
<td></td>
<td>es.vt</td>
<td>es.vt</td>
<td>TimeInterval</td>
</tr>
<tr>
<td>es.loc</td>
<td>es.loc</td>
<td></td>
<td>es.loc</td>
<td>SpaceInterval</td>
</tr>
</tbody>
</table>
FIGURE 1 A UML diagram for dynamic segmentation (adapted from 6).
FIGURE 2 Extended object types.
FIGURE 3 Overview of the Singapore road network.
FIGURE 4 An example representation of events on a route.