

OBJECT-RELATIONAL FEATURES FOR MODELING AND ANALYSIS OF SPATIO-TEMPORAL DATA

S. Scheugenpflug, M. Schilcher

Technische Universität München, Fachgebiet Geoinformationssysteme, D-80290 München, Germany
(scheugenpflug, schilcher)@bv.tum.de

TS WG IV/1

KEY WORDS: GIS, Databases, Modeling, Analysis, Ecosystems, Temporal, Monitoring

ABSTRACT:

At Technische Universität München extensive official, forestal, climatological, touristical geodata and metadata on the unique ecosystem Bavarian Forest National Park were gathered since 1996 to set up a GIS-platform used for research projects as well as for education. To improve data availability of this precious datapool the GIS-platform has been Web-enabled by migrating all its data content to a new integrated and highly scalable 3-tier GIS-environment, using ArcInfo, ArcSDE and Oracle(Spatial). Currently seven applications of different university chairs and institutions access this central geodata server for reading and writing, guided by metadata. The objective is to build an interdisciplinary sustainable GIS-platform with unique data on analyzing the ecosystem's short and long-term natural processes like its recovering behaviour and forest development after bark beetle outbreak. In this manner, the datapool is updated and extended automatically through its distributed write-access concept in an interdisciplinary network.

The main objectives of this paper are new aspects of using object-relational features of the underlying database management system Oracle to improve the extensibility and flexibility of data models, enhance interoperability and analyzing potential as well as to ensure consistency by defining standards based on abstract data types. The potential is demonstrated considering two examples:

First the scenario of deadwood spread due to bark beetle outbreak during 1994 and 2002 as well as spatio-temporal analysis of forest rejuvenation, that is all hope of getting back to a green, more beetle-resistant forest in future. The object-relational extension of data models can be combined with data mining techniques to analyse e.g. spreading patterns of bark beetle outbreak. The vision is to merge data that represents factors which influence bark beetle activity and to derive conclusions about the correlation of these factors. The second example describes the application of real time deer tracking based on object-relational features in Bavarian Forest National Park.

ZUSAMMENFASSUNG:

An der Technischen Universität München wurde seit 1996 für den Nationalpark Bayerischer Wald ein umfangreicher GIS-Datenbestand aufgebaut, der u.a. amtliche, forstliche, klimatologische und touristische Geodaten über das einzigartige Ökosystem des Nationalparks Bayerischer Wald umfasst und seither für Forschungsprojekte, die GIS-Lehre und in der Praxis genutzt wird. Um die Datenverfügbarkeit und damit den Nutzen dieses kostbaren Datenpools zu verbessern, wurden sämtliche Daten in eine internetfähige 3-tier GIS-Systemplattform auf Basis von ArcInfo, ArcSDE und Oracle(Spatial) migriert. Derzeit greifen sieben verschiedene Applikationen, unterstützt durch Metadaten, lesend und schreibend auf den Geodatenserver zu und tragen durch die Einbringung von Analyseergebnissen zum Aufbau und zur Erweiterung einer nachhaltig und interdisziplinär nutzbaren GIS-Plattform bei, die einzigartige Daten zu unmittelbaren und langfristigen Prozessen dieses Waldökosystems wie etwa die natürliche Waldentwicklung nach Borkenkäfermassenvermehrung archiviert.

Kern dieses Papers ist der Ansatz, objektrelationale Modellierungskonstrukte und Methoden des ORDBMS Oracle der GIS-Plattform zu nutzen, um flexiblere, erweiterbare und konsistenzsichernde Datenbankmodelle zu generieren und das Analysepotential zu erhöhen. Der raumzeitliche Analyseansatz wird anhand zweier Beispiele demonstriert:

Erstes Anwendungsbeispiel ist die Totholz Ausbreitung nach Borkenkäferbefall von 1994 - 2002 und die natürliche Waldverjüngung, auf der die Hoffnung auf künftige Entwicklung eines gesunden, Borkenkäfer-resistenteren Waldes ruht. Die Erweiterung von Datenbankmodellen um objektrelationale Konstrukte kann mit Methoden des Data Mining kombiniert werden, um beispielsweise das Ausbreitungsmuster des Borkenkäfers zu analysieren. Ziel ist die Zusammenführung und Abschätzung der Einflussfaktoren, die für die Borkenkäferausbreitung entscheidend sind. Das zweite Beispiel beschreibt die Anwendung der real time Hirschverfolgung im Nationalpark Bayerischer Wald auf Basis objektrelationaler Datenbankstrukturen.

1. MOTIVATION AND BACKGROUND

1.1 Bark beetle outbreak in Bavarian Forest National Park

Bavarian Forest National Park, founded in 1970, with its 243 square kilometers in size is part of the biggest contiguous woodland of middle europe. In this unique forest ecosystem,

located at the border triangle of Germany, Czech Republic and Austria, nature is left on its own as any anthropogenic actions are prohibited in the park's centre zone. A big variety of protected native plants and animals can be found in that ecosystem. The park's idea of self-regulation however also allows varmints like the bark beetle to spawn and spread within national park borders. Insect calamities have always occurred

even in past centuries, especially after big windthrows. Broken trees build natural origins of a new beetle generation that is able to reproduce two times a year and therefore is able to spread rapidly in its beloved spruce tree paradise.

Therefore people differ on the idea of a german jungle, especially if they are owner of private forests adjacent to national park borders. Thus the main focus is on analyzing how the national park's vegetation will develop in the near future resp. how long it will probably take to get back to a green forest which matches better with human inherent expectations of what a national forest should normally look like.



Figure 1. Deadwood in Bavarian Forest National Park

Trying to find answers on that question boils down to understand the spreading behaviour resp. the impact of influencing factors as well as to monitor the development of natural rejuvenation. This is the only way and therefore represents the hope to put oil on troubled waters and to get back to a green, more beetle-resistant forest in future, at least for our next generation.

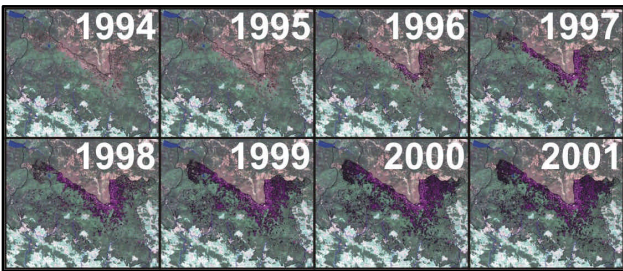


Figure 2. Time series of deadwood spreading after bark beetle outbreak

In the meantime almost 4000 ha of spruce stands are died off which makes up roughly one third of the core park area, where no protection measures are tolerated. The bark beetle's behaviour is not yet entirely explored and still under research. Though it is no longer a question when and how far bark beetles fly and that main influencing factors of the beetle's activity can be found in climate, forest structure including tree species and age, soil type resp. water situation, yet dead wood spreading seems to be rather arbitrary. It is hard to find regular patterns that match fine with the site-related-factors mentioned above. In addition all real world natural phenomena are particularly tough to see through and to predict because nobody knows what kind of e.g. spring temperatures we'll face in near future. Currently a further completely different approach is followed that lights up the spreading from another point of view by capturing and analyzing the air's pheromon accumulation to get to an entire

understanding about the beetles' communication and dispersion behaviour.

1.2 Natural forest rejuvenation carries hopes

Monitoring the very young forest is time consuming as it cannot (yet) be automated by using new techniques like laserscanning or radar. Considering the shielding of older trees there's no way to obtain complete and reliable information about the number and condition of upcoming plantlets or natural competition among the vegetation. This leads to get down on the floor, setup long-term sample areas, survey every single sapling, mark them out using unique identifiers and reexplore them at least several consecutive years!



Figure 3. Natural rejuvenation raises hopes for forest recovering

In Bavarian Forest National Park a bunch of long-term monitoring areas were selected in a manner to represent different area types with varying impact characters, to allow distinctions between planar and exposed monitoring areas, certain degrees of matured forest dying off as well as different National Park regions that differ in accepting man-made reforestation or not. Figure 4 documents one of the 40 x 40 m monitoring squares that comprises border area, core zone, plant sociological surveying areas, sample circles and soil samples extraction points.

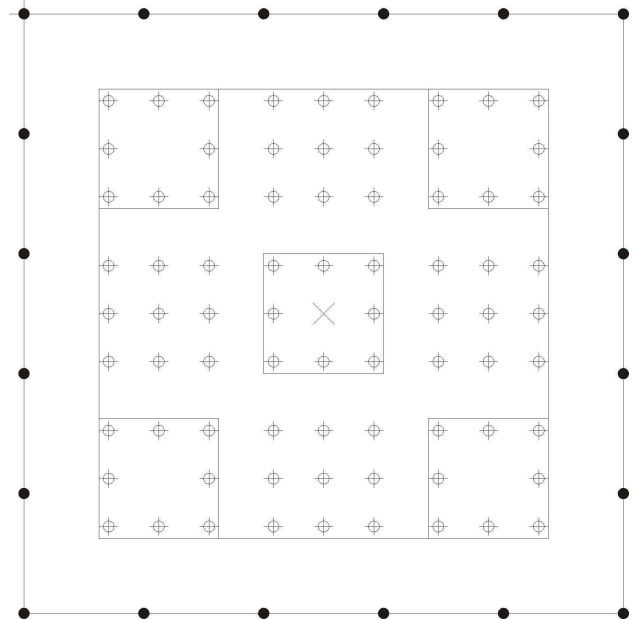


Figure 4. Sketch of a monitoring area to analyze forest rejuvenation development

From 1998 to 2000 lots of data on those monitoring areas were gathered to find answers to the question whether there are enough seedlings to assure reforestation after the devastating bark beetle infestation and dying of lots of forest stands. Besides seedlings all other kinds of vegetation in the test areas were included into data acquisition to allow an estimation of growing conditions and vegetative competition (Bauer, 2002). The data was statistically evaluated using proved methods, models and hypotheses. E.g. the spatial distribution of events was calculated using the CLARK & EVANS-Index, that gives the relation between measured and expected distance to the nearest neighbour. It is a measure for regular or clustered horizontal distribution. A clustered distribution will give values below 1 whereas a complete hexagonal distribution produces the maximum value of 2,15.

So far no analysis of this rejuvenation data has been performed using GIS and database technology, particularly in regard to new possibilities provided through object-relational features and data mining. Since database models get closer to real world objects and events by allowing the setup of an integrated and adjusted representation based on user-defined types, methods binded to database objects, and further declarative information that resides in XML, a complete description of spatial objects in database systems can be provided for the first time. Therefore it is obvious to make use of object-relational structures provided by the underlying DBMS to develop a new approach to examine bark beetle spread and forest rejuvenation. In the following a brief introduction of object-relational modeling concepts is given. As the implementation is based on an ORACLE 9i database, the corresponding SQL-dialect is followed.

2. OBJECT-RELATIONAL FEATURES

Within the classic relational database model there are only scalar but no complex data types. With the introduction of **object types** the definition and composition of abstract data types is possible. An abstract data type can be comprised of a multitude of scalar types and again of user-defined complex types and enhances consistency when creating database models.

```
create or replace type POSITION_TY as object (
    x    NUMBER (9,2),
    y    NUMBER (9,2)
);
```

```
create or replace type TREE_TY as object (
    position    POSITION_TY,
    species     VARCHAR2 (30),
    plantation_date DATE
);
```

Data types only represent descriptions of data structures. To ensure persistency a table must be bound to the data type. A relation with column objects can be created that is a set of young trees with tree ids as:

```
create table TREE_TAB (
    tree_id    INTEGER,
    tree       TREE_TY
);
```

In addition to representing column objects object types can also be used to define row objects that can be managed using **object tables**:

```
create table TREE_OT of TREE_TY;
```

Object views allow users to treat relationally data as objects as they allow to synthesize objects from data that continues to be stored in relational tables. Object Views are therefore often referred to as “natural bridges” between both paradigms. Object views have similar functionality like object tables. They can have methods, belong to collections, reference one another, have object identity and can be accessed from SQL. In addition tables that are assigned to object views can be updated by using special *instead of triggers*.

Any instance (row) of an object class contains a unique ID called object-ID (**OID**). Generally OIDs are system-generated but can also be derived from a primary key column or can be user-defined.

Relationships between objects can be defined using **reference types**. A reference column stores OIDs of associated (row) objects since column objects do not have inherent OIDs and therefore cannot be referenced. Row objects that belong to a reference can be selected and dereferenced using the Deref resp. VALUE operator. Modeling object relationships with OIDs and REFs is often compared with foreign key relationships inside the relational model but implicates some benefits like the ability to distinguish between equal and identical objects. Objects are identical if they have one common OID. They are equal if they have different OIDs but coincide with their attributes and values.

In the classical Entity-Relationship-Model aggregations and compositions are modelled through master-detail-relationships. Object-relational dbms provide **collection types** that contain multiple elements and thus are suitable to express 1:n relationships directly. Each element or value for a collection has the same substitutable data type. The most popular collection types are varrays and nested tables.

A **varray** contains a variable number of ordered elements. Varray data types can be used as a column of a table or as an attribute of an abstract type.

Named table types can be created in an Oracle database using SQL. These table types can be used as **nested tables** to provide the semantics of an unordered collection. As with varray a nested table type can be used as a column of a table or as an attribute of an object type.

```
create type TREE_NT as table of TREE_TY;
```

Multi-Level-Collections that lead to multiple nested tables can be realized if useful for applications but it's up to the user to balance – a more intuitive representation of data vs. higher complexity of accessing the data.

An object type declaration can also include **methods** that are defined on values of that type. When using these object types in tables their methods are also applied to the data of these tables. The method is declared in the *create type* statement and the code for the function itself (the definition of the method) is in a separate *create type body* statement.

```

create or replace type TREE_TY as object (
  position      POSITION_TY,
  species       VARCHAR2 (30),
  plantation_date DATE,
  member function AGE (plantation_date in DATE)
  return NUMBER,
  pragma restrict_references (AGE, wnds));

```

The keyword *pragma* and its bounded clause is mandatory if the method AGE is to be used in queries as it says the AGE method will not modify the database (WNDS = write no database state).

```

create or replace type body TREE_TY as
  member function AGE (plantation_date DATE)
  return NUMBER is
  begin
    return round (SysDate – plantation_date);
  end;

```

The following query executes the age method in the data type tree_ty and returns the age of all tree objects in relation TREE_TAB. Values of components of an object (attributes and methods) are accessed with the dot notation.

```

select tree_id, t.tree.AGE (tree.plantation_date)
from TREE_TAB t;

```

When using methods there is no need to store variable data - such as the age of planted trees for the purpose of rejuvenation outside the park's centre zone. Instead static data (such as the plantation_date) can be accessed to derive variable data via functions and procedures.

Object types can be mapped to the prevalent OO languages like C++ and Java. Thus object type instances in the database can be accessed and modified to and from C++ resp. Java applications (Lee, 2003).

Type inheritance allows sharing similarities among data types as well as extending their characteristics. With single type inheritance a type may extend (inherit from) one supertype. Such a type (called subtype) inherits all its supertype's attributes and methods. A subtype may also add new attributes and methods and / or override inherited methods.

The root type of a hierarchy must be declared to be not final:

```

create or replace type TREE_TY as object (
  position      POSITION_TY,
  species       VARCHAR2 (30),
  member function STOCK () return NUMBER) not final;

```

A subtype can be created under a non final type. It inherits all attributes and methods from its supertype. It can add new attributes and methods and / or override inherited methods.

```

create type CONIFER_TY under TREE_TY (
  status_of_damage VARCHAR (20),
  height           NUMBER (4,2),
  member function AGE () return NUMBER,
  overriding member function STOCK () return NUMBER);

```

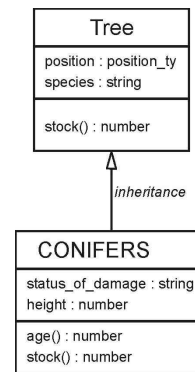


Figure 5. Type Inheritance

Oracle Spatial and Oracle Locator, both core features of the Oracle database provide a natively supported **open vector data type** *SDO_Geometry* for storing vector data and a set of spatial operators and functions to perform spatial analysis inside the database. The following command retrieves the topological intersection of deadwood and stand features and writes the corresponding geometry into a new spatial table:

```

insert into result_table (id, geom)
  select d.id, sdo_geom.sdo_intersection (d.geom, m.diminfo,
    s.geom, m.diminfo)
from DEADWOOD d, STANDS s, user_sdo_geom_metadata m
where m.table_name like 'stands'
and m.column_name like 'geom'
and d.id = 324 and s.id = 5141524252;

```

In the meantime the *SDO_Geometry* data type has become a de facto industry standard that is supported by all major GIS vendors' products to push interoperability. R-tree and quadtree indexes can be used standalone or in conjunction. Each index type is appropriate for different situations, although R-tree indexing is often the best choice because of its capacity to operate directly against geodetic data (Geringer, 2003).

In the current version Oracle database 10g native data types for storing raster and persistent topology data were added to further extend possibilities to model real world objects based on object-relational database structures.

The completeness and therefore the quality of spatial features' descriptions can be further enhanced with the introduction of a new system-defined datatype **XMLType** that can be used as the datatype for columns in tables and views to create, extract, and index XML data. A feature's position and spatial representation is then modeled by using vector resp. raster types, its characteristics are captured by (complex) attributes and any further information on the spatial object, that cannot be structured to be kept in the data types mentioned, is maintained in xml-documents, all stored natively within the database.

For modeling highly dynamic changes of real world objects the SQL type *TIMESTAMP* can be used which is a high-precision **time and date type** that allows storing fractions of a second (Qian, 2004). The timestamp values are basically points on a linear time axis.

For slower changes in state we can use *DATE* or create own adapted abstract data types that can express discrete dates or times intervals. The data types can be flexibly extended to meet an application's accuracy requirements and provide attributes to express the fuzziness of a time record, for instance if a time

interval cannot be determined as accurate as the data type could store (Plabst, 2001).

```
create or replace type POINT_IN_TIME_TY as object (
  year    NUMBER (4,0),
  month   NUMBER (2,0),
  day     NUMBER (2,0),
  hour    NUMBER (2,0));
```

```
create or replace type TIME_TY as object (
  start    POINT_IN_TIME_TY,
  end      POINT_IN_TIME_TY,
  start_reliability  NUMBER (1,0),
  end_reliability    NUMBER (1,0));
```

3. OBJECT-RELATIONAL APPLICATION DEVELOPMENT

In terms of Bavarian Forest National Park, object-relational database technology can be used to model different applications, e.g. the spatio-temporal process of deadwood spreading based on its influencing factors or real time deer tracking.

For the **development of bark beetle outbreak** some major influencing factors have been proved to be crucial.

After windthrows climatological aspects are considered to be decisive to trigger off a bark beetle calamity. The year's first day at 20 degrees above zero, the number of consecutive days at that temperature level and the last day of snow coverage are important to describe the beetles' spring swarming potential and the likelihood to breed twice a year (Nationalpark Bayerischer Wald, 2001). These climate measurements affect the cambial temperature of trees and the temperature sum which on their part affect the beetles' development conditions. The climate parameters are surveyed up to three times a day at 14 weather stations located in the national park region and kept in a relational database model. We can create object types based on the structure of these relational tables for any weather station:

```
create or replace type WALDHAEUSER_TY as object (
  year          NUMBER (4),
  month         NUMBER (2),
  day           NUMBER (2),
  t_max        NUMBER (3,1),
  t_min        NUMBER (3,1),
  t_kw         NUMBER (3,1),
  snow_cover   NUMBER (3));
```

Based on this type an object view is created and OIDs are assigned to the climate datasets of the corresponding weather station per year:

```
create or replace view WALDHAEUSER_1993_OV
of WALDHAEUSER_TY
with object identifier (year, month, day) as
select year, month, day, t_max, t_min, t_kw, snow_cover
from WALDHAEUSER
where year = 1993;
```

The columns of the relational base tables (weather stations) are now accessible as row objects through their corresponding object views. Any detail tables of the underlying relational model could now be simulated by further object views with references that could point to the row objects of the particular upper object view, to allow accessing the data in either way.

The next step is to create a superior type CLIMATE_TY, that keeps references to the row objects of all weather stations. The data types are all bound not to object tables but to object views as shown beforehand:

```
create or replace type CLIMATE_TY as object (
  waldhaeuser_per_year ref    WALDHAEUSER_TY,
  lusen_per_year       ref    LUSEN_TY,
  taferlruck_per_year  ref    TAFERLRUCK_TY);
```

To aggregate deadwood spreading influencing factors per year in a main table (DEADWOOD), that addresses all corresponding data stored in any existing (relational) database model of the GIS-platform, table types are created based on abstract data types, that represent each factor like in this first example, CLIMATE_TY:

```
create type CLIMATE_VALUES_NT
as table of CLIMATE_TY;
```

```
create table DEADWOOD (
  year          NUMBER (4),
  geom          MDSYS.SDO_GEOMETRY,
  climate       CLIMATE_VALUES_NT,
  stand        STAND_VALUES_NT,
  soil         SOIL_VALUES_TY,
  object_info  XMLTYPE)
nested table climate store as CLIMATE_VALUES_NT_TAB,
nested table stand store as STAND_VALUES_NT_TAB,
nested table soil store as SOIL_VALUES_NT_TAB;
```










time NUMBER (4)	geom SDO_GEOMETRY	climate CLIMATE_VALUES_NT	stand STAND_VALUES_NT	...	object_info XMLTYPE
1994					
1995					
1996					
...					

Figure 6. Main table for analyzing bark beetle spreading

Deadwood polygons per year are derived from CIR images that are acquired during annual flight campaigns each summer. As they document the increase of deadwood areas since the preceding year - so to speak they capture the bark beetles' work with one-year delay, they have to be correlated with the climate values of the previous year. This is achieved by referencing climate row objects of object views of the preceding year. The deadwood polygon geometries are stored in a spatial column.

As for data concerning stand structure information like species and age of trees resp. soil types that are important to model the humidity penetration of the ground, one can follow suit whereby this kind of data is less dynamic or even quasi-static.

Any further information in respect of the situation of infestation that cannot be structured to be stored as attributes can be kept and queried directly inside the database in terms of document-resp. data-centric XML (Lee, 2003).

Currently **rejuvenation data** gathered on the described monitoring areas from 1998 to 2000 is modeled and integrated to be analyzed in combination with existing inventory data of 1996 and 2000 using GIS and database oriented spatial functions. In addition data mining techniques are explored to be applied to find patterns in data that prove hypothesis how rejuvenescence will develop in near future.

Another imaginative application to make use of object-relational features is **real time tracking of deer location**. Concerned parties can be notified when deer are crossing borders or other areas of interest. The total national park area or the upper area 1100 m above sea level can be entered into a database table:

```
create table AREA_OF_INTEREST (
    area_id      NUMBER (2),
    area_name    VARCHAR2 (50),
    border       MDSYS.SDO_GEOMETRY
);
```

Deer migration can be followed – based on GPS collars - by tracking its locations at a fixed interval (e.g. every few seconds), and in real-time inserting the records into another table:

```
create table DEER_LOCATIONS (
    deer_id      NUMBER (3),
    time         TIMESTAMP (2),
    location     MDSYS.SDO_GEOMETRY
);
```

In this table time is stored using the SQL type **TIMESTAMP**, which is a high-precision time and date type that allows to store fractions of a second (Qian, 2004). Whenever a GPS receiver updates its location the current timestamp and location will automatically be sent out and inserted into the above table:

```
insert into DEER_LOCATIONS values (3,
    to_timestamp ('17-APR-2004 19:17:21.00',
    'dd-mon-yyyy hh24:mi:ss.ff'),
    mdsys.sdo_geometry (2001, 82032,
    mdsys.sdo_point_type (4607260.47, 5425484.18, null),
    null, null)
);
```

The query inserts a new record for the location of deer with id 3 at the specified timestamp. It is possible to insert thousands of such records in one second which makes it quite feasible for real time tracking of numerous deer.

Now a database trigger can be implemented that is sparked off whenever a new record is inserted into a specific table. In the trigger body it can be checked if deer is still within the national park resp. within another area of interest. This operation can be achieved using the *relate function*. The query from the trigger body to determine if deer is still within the park is:

```
select sdo_geom.relate (a.location, 'ANYINTERACT',
    b.border, 0.5)
from deer_locations a, area_of_interest b
where a.deer_id = 3 and b.area_name like 'nationalpark' and
a.time = (select max (time) from deer_locations where deer_id =
3);
```

The result of the query is 'TRUE' if the deer is still within park's borders or 'FALSE' otherwise. The subquery returns the most current timestamp for the deer with id 3. If a 'FALSE' value is returned the trigger can e.g. open an HTTP connection and send a message to a Web service outside the database with the deer id, the location and the time the deer is crossing the border. The Web service can then act accordingly, for example sending out emails to the concerned institution or creating dynamic maps with the current deer location.

Let us assume that the deer perambulate special areas once a year we can create user-defined object types that represent such events as spatio-temporal zones:

```
create or replace type ST_ZONE_TY as object (
    name          VARCHAR2 (50),
    from          DATE,
    till          DATE,
    zone          MDSYS.SDO_GEOMETRY);
```

A method can be added to the object type that validates if deer location and stopover falls inside such a zone. By this means histories and changes to real world objects can be captured and managed inside the database.

4. RESUME AND CONCLUSION

Object-relational technology in DBMS helps to model and analyze spatial-temporal events as it provides highly flexible means for storing, manipulating and validating diverse and complex data structures. As for applications in Bavarian Forest National Park the real challenge is to understand the relationships of the most complex processes at all - those of nature - and find definite and describable entities that can be part of a database model. Thus it is often useful to extend the concept of objects from a conventional point of view to a more general sense, so to speak to describe real world scenes (instead of objects) that show on their part some correlation with a set of commonly used objects. Object views leave existing (relational) data models unchanged but can centrally merge miscellaneous information of any database model and serve for making use of object-relational features in spatio-temporal applications.

References from Books:

Loney, K., 2003. *ORACLE9i: Die umfassende Referenz*. Carl Hanser Verlag, München Wien, ISBN: 3-446-22170-0

References from other Literature:

Bauer, M., 2002. *Walddynamik nach Borkenkäferbefall in den Hochlagen des Bayerischen Waldes: Dissertation at Lehrstuhl für Waldbau und Forsteinrichtung*, Technische Universität München

Geringer, D., 2003. *Oracle Spatial Best Practices: An Oracle Technical White Paper*, Dezember 2003

Lee, G., 2003. *Simple Strategies for Complex Data: Oracle 9i Object-Relational Technology: An Oracle Technical White Paper*, October 2003

Nationalpark Bayerischer Wald, Bayerische Staatsforstverwaltung, 2001. *Waldentwicklung im Bergwald nach Windwurf und Borkenkäferbefall: Academic series book 14*, July 2001

Plabst, S., 2001. *Entwicklung eines objektrelationalen Datenmodells für ein kulturhistorisches Geoinformationssystem: Thesis at Fachgebiet Geoinformationssysteme*, Technische Universität München

Qian, L., 2004. *Oracle Database 10g – Developing Spatial Applications Using Oracle Spatial and Map Viewer: An Oracle Technical White Paper*, February 2004

References from websites:

General Oracle technical resources: <http://otn.oracle.com>

Oracle Spatial resources: <http://otn.oracle.com/products/spatial/>