OSHADHI : A Biodiversity Information System

R. Vidya † *   Jayant R. Haritsa ††

† HCL HP Limited (Research & Development)
50–53 Greams Road, Madras 600006, India
{rvidya}@indus.hclhprnd.uunet.in

†† Dept. of Computer Science and Automation
Indian Institute of Science, Bangalore 560012, India
{haritsa}@csa.iisc.ernet.in

Abstract

The economic importance and uses of the large number of plant species of India make it essential for their biodiversity to be conserved. In order to undertake the steps for biodiversity conservation such as identification of species and monitoring of climatic conditions, it is essential to efficiently manage the vast amount of biodiversity related data. This data includes taxonomic data, geographic data, bibliographic data and experts data. In this paper, we present an object-oriented model for biodiversity databases, and describe the design and implementation of OSHADHI, a biodiversity information system that implements the proposed model. OSHADHI implements a variety of query-specific access methods for efficient query processing.

1 Introduction

The genetic plant base of India is represented by a large number of plant species which include "local" species (species that grow naturally in the country) and "foreign" species (species that have been brought in from various parts of the world). During the past

*This work was done while the author was with the Dept. of Computer Science and Automation, Indian Institute of Science.

decade, several new local species have been identified and there has also been a rapid growth in the number of foreign species. The rich economic potential and uses of these plant species make it essential for their *biodiversity* to be conserved. Several measures for biodiversity conservation have been suggested by researchers and scientists in the area of environmental sciences. These measures include identification of species, monitoring the status of endangered species, and monitoring the climatic conditions of various geographic locations.

Several organizations all over the country are working on the implementation of the biodiversity conservation measures. One of the major hurdles faced by conservation scientists and researchers is the *management* of the vast amount of biodiversity related information that is available (there are more than 250,000 documented plant species !). This information includes data on species characters (e.g., flower, fruit, habitat), endangered species, species of medicinal value, climatic conditions (e.g., temperature, rainfall), soils and land use, geographic distribution of species, botanical gardens and herbariums [1], bibliographic references for species and geographic regions, experts in related fields, and a host of other information.

Broadly speaking, the above biodiversity data may be classified into four groups: the *taxonomic* data (details of all the species, their classifications and characters, etc.), the *geographic* data (details of the geographic regions, their environmental conditions, etc.), the *bibliographic* data (details of the journals, articles and textbooks in the field of biodiversity) and the *experts* data (details of the experts in the area of biodiversity, their affiliation, etc.). An important point to be noted is that the data in these groups are both *intra-related* and *inter-related*. For example, the geographical distribution of a species relates the taxonomic data and the geographic data. Similarly, species may participate in predator, symbiotic or parasitic relationships with each other. A high-level organization of the biodiversity database is shown in Figure 1.

Apart from the need to aid the process of biodiversity conservation, maintenance of information such as whether a species is local or foreign is also essential in light of the recently concluded world treaty, *General Agreement on Trade and Tariffs (GATT)*, which places emphasis on property rights.

From the above discussion, it is clear that the biodiversity domain requires data management support. One possible approach is to use the popular relational database management systems, which have proved highly effective for commercial applications such as banking, inventory control and airline reservation. However, they are not well suited for recent complex and advanced database applications such as geographic information systems, computer aided design systems, and network management systems – biodiversity data management also falls into this category of applications that are not amenable to relational DBMS. This is because biodiversity (and the other) applications differ significantly from the commercial applications in both their *modeling* and *processing* requirements. The following points highlight some of the requirements of the biodiversity application which are not satisfactorily satisfied by relational database systems:

(i) Many of the objects dealt with in the biodiversity application are complex. For example, the taxonomy of species is a four level hierarchy *Order*, *Family*, *Genera*, *Species*.

[1] Herbariums are repositories where collections of dried or preserved plants (or their parts) are archived.

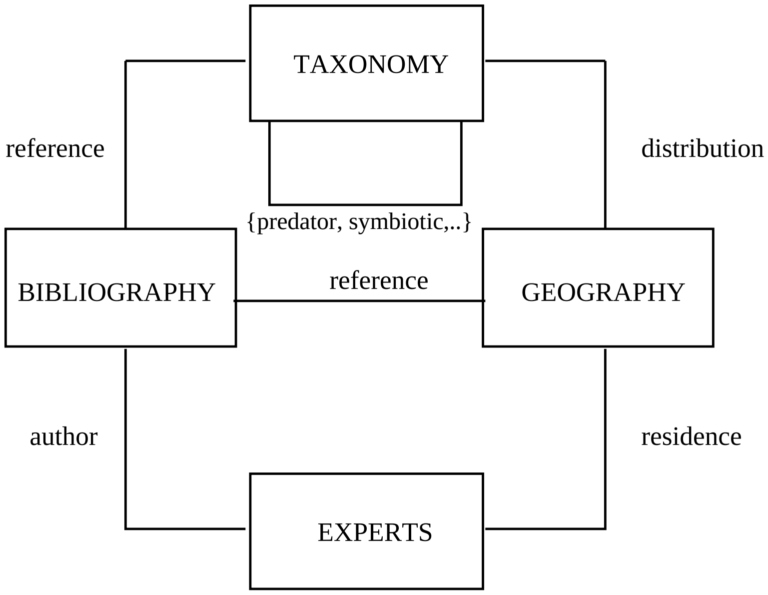

Figure 1: The Biodiversity Data Organization

Similarly, the flower character of a plant species is an aggregation of sepal characters, petal characters, inflorescence, etc. A similar aggregation is also exhibited by the spatial data (polygons, lines, etc.) necessary for representing geographic regions.

(ii) Activities such as monitoring of climatic conditions and endangered species population in a region require efficient version control mechanisms.

(iii) The biodiversity application calls for changes in the schema with time. For example, schema modification such as adding new character types, taxonomy levels, spatial and geographic object types should be possible without affecting the structural consistency of the database and without having to recode the application.

(iv) Some of the objects in the biodiversity domain possess certain unique features that may be exploited to build query-specific access methods. For instance, the descriptive characters of species usually have small domains in terms of their values (typically five to ten) and this feature may be used to provide a specialized index for these characters. Similarly, index structures such as R-Tree [Gutt84] or Quadtree [Same89] may help in improving the access time for spatial data. Therefore, it is essential to be able to add on access methods as the system evolves.

(v) Queries such as displaying the spatial distribution of species and the climatic condition over a specified region require identifying the polygons representing the distribution that are contained inside the polygon representing the geographic area.

The OSHADHI System

Recent research in the field of databases has resulted in the emergence of new approaches such as the *Object Oriented Databases* which support the desired features to cater to the requirements of the advanced applications. In this paper, we present the design and implementation of **OSHADHI** [2], an object oriented biodiversity information system. OSHADHI is a working prototype system (13,000 lines of C++ code) that has been built using the well-known **EXODUS** (Extensible Object Oriented Database Management System) [Care86a] system as the back-end. The OSHADHI architecture is composed of three major components, namely, the *EXODUS Storage Manager*, the *OSHADHI Database Server* and the *OSHADHI Graphical User Interface*.

Users of OSHADHI interact with the system through the Graphical User Interface, which provides user-friendly customized forms for entering bio-diversity data and queries. This information is passed on to the Database Server which parses and processes the information. The server implements the OSHADHI object model and uses the EXODUS Storage Manager to store the database. It is written using the E language [Rich89], which is a database programming language supported by EXODUS. The EXODUS Storage Manager, which forms the lowest layer of OSHADHI, serves requests from the database server with regard to object management and transaction processing.

The OSHADHI database includes the taxonomy details of various species, their characters, classifications and textual descriptions. It also has the details of various geographic regions and their climatic conditions. The OSHADHI system handles queries such as the identification of species given a set of identifying characters (e.g., flower, fruit, or leaf), viewing the characters of a species, listing species with specific characters in a geographic location, viewing maps of geographic regions, etc. For efficient processing and fast access of data, query-specific access methods such as B^+tree indexing [Baye72], Quadtree indexing [Same89] and pointer intersection [Kort91], are incorporated in OSHADHI.

The OSHADHI data model has been developed based on the specification for the plants of Western Ghats, provided by the *Centre for Ecological Sciences* at our institute. Information about these plants is currently stored in the prototype database system.

Related Work

Several projects have been undertaken by various organizations, both in India and abroad, with the aim of addressing the biodiversity data management problem. For example, *INMED PLAN (Indian Medicinal Plants National Network of Distributed Database)* is a database built for the medicinal plants of India. Other related projects include the Madras-based *Mangrove Ecosystems Information Service* project, the Australian *ERIN (Environmental Resources Information Network)* project, the British *PLATO (PLAnt TOxins)* project [Poun95], and the Australian *INTKEY* project.

Most of these projects have been built using either relational database systems such as ORACLE, or using other technologies such as Hypermedia. As discussed above, relational database systems may not be the best choice for biodiversity database systems. While

[2] OSHADHI is the Sanskrit word for plant kingdom.

hypermedia technology would be suitable to navigate through various related objects, it may not be adequate to handle queries such as identification of species based on their taxonomic and geographic characteristics. In addition, these projects usually address only a subset of the problem: The PLATO and INTKEY projects use a database in the DELTA (DEscriptive Language for TAxonomy) format [Poun95]. While DELTA is a free format text which is flexible, its application is limited to taxonomy and identification. In contrast, OSHADHI tries to provide a comprehensive solution to the biodiversity data management problem.

Organization

The rest of the paper is organized as follows: In the following section, the biodiversity application is described in terms of the real world data objects that need to be modeled, the relationships between them, and their constraints. In Section 3, the modeling and processing requirements of the biodiversity application are discussed. An object model to satisfy these requirements is proposed in Section 4. The system architecture of OSHADHI is described in Section 5, and in Section 6, the various access methods that are used for query processing in OSHADHI are presented. Section 7 discusses the various types of queries and their processing in OSHADHI. A brief description of the OSHADHI graphical user interface is presented in Section 8. Finally, in Section 9, we present the conclusions of our study and outline future research avenues.

2 Biodiversity Information

As mentioned in the introduction, biodiversity data can be classified into four major groups – the taxonomic data, the geographic data, the bibliographic data and the experts data. In this section, we describe each of the groups and the relationships between them.

Taxonomy Data

Taxonomy is a study aimed at producing a classification of plant species that reflects the similarities and differences among the species. Each unit of classification is called a *taxon*. The four major taxons in the classification of species are *Order*, *Family*, *Genera* and *Species*. An order contains several related families, which means that families that exhibit certain similar characters are grouped into an order. Likewise, a family is represented by a collection of genera and a genera is a collection of related species. Each taxon has a biological name. For example, the common "champa" plant belongs to the Order *Ranales*, Family *Magnoliaceae*, Genera *Michelia* and Species *Champaca*. Species are typically identified based on their *leaf, stem, flower* and *fruit* characters. Hence, these characters are referred to as *identifying* characters. There are other characters, for example, the *root* character, which are usually not used in the identification of a species. These characters are referred to as *non-identifying* characters.

A flower is composed of several parts, namely, *petals*, *sepals*, *inflorescence*, *carpels*, *ovaries*, *anthers*, *style* and *stigma*. Each of these flower-specific characters could be used for identification of a species and hence they are also identifying characters. Similarly, a leaf contains parts such as blades, sheath, etc. In summary, every order, family, genera and species is associated with a set of identifying and non-identifying characters that differentiate it from the other orders, families, genera and species, respectively.

In addition to the identification information, a species may have synonyms (multiple names, usually in different languages), line drawings and a large amount of textual information associated with it. The textual information may include extracts from journals and textbooks.

The taxonomy data exhibits several intra-relationships. For example, two species might have a *parasitic*, *symbiotic* or *predator* relationship between them.

Geographic Data

There are essentially four types of geographical organizations – the *bio-geographic provinces* (e.g., Malabar, Coromandel), the *biomes* (vegetative regions, e.g., evergreen, deciduous), the *administrative organizations* (e.g., district, state), and the *forest reserves*. These organizations, along with the botanical gardens and herbariums, represent the geographical data. Every geographical region has a map associated with it. The environmental conditions (e.g., soils, temperature, pressure, humidity) in these regions need to be recorded in two forms. One is the *spatial* (i.e. regional) variation of the condition and the other is the *temporal* (i.e. over time) variation of the condition.

There is a rich inter-relationship between the geographic data and the taxonomic data. The distribution of a species in a geographical organization is an example of this type of relationship. Another instance is that, for all the species found in a geographic region, the spatial and temporal variation in population have to be recorded.

Bibliographic and Experts Data

The bibliographic data includes abstracts/references of biodiversity related journals and textbooks. A journal has a collection of articles which refer to species or geographical regions. Each article has a group of authors.

The experts database includes information about the experts in the area of biodiversity, their area of specialization, their affiliation, and their list of publications.

Although our data model (Section 4) includes all of the above four databases, we focus primarily on the taxonomic and geographic databases in the remainder of this paper since these databases are the main components of the biodiversity domain. Our examples are also, therefore, mostly taken from these databases.

3 Requirements of the biodiversity application

Having seen the description of the biodiversity database in the previous section, we now move on to con6 mtring the modeling and processing requirements of the biodiversity application.

3.1 Modeling requirements

1. The description of the biodiversity database in the previous section reveals several *aggregation* hierarchies. For instance, the taxonomy of species results in a four level aggregation of order, family, genera and species. These aggregations must be supported by the model. Apart from the taxonomic aggregation, there are several other aggregations in the biodiversity database. The details of these other aggregations are discussed in the object model (Section 4).

2. As discussed earlier, every taxon is associated with a set of identifying characters (such as flower, fruit, leaf, or seed). The set of characters that identifies a taxon may vary from one taxon to other. In other words, order A may have a subset of flower characters as its identifying characters, while order B may have a subset of leaf characters as its identifying characters. For the order, family and genera taxons, maintaining each of the possible identifying characters as an attribute would result in several null fields since the number of characters that can be used for identification is large (approximately 100), while the actual number of identifying characters associated with these taxons is small (typically of the order of 10 to 20). This problem does not arise for species since they have values for *all* the identifying characters.

   The information about the distribution of a species could be available at any geographical organization level (bio-geographic provinces, biomes, administrative organizations, or forests). For example, we might have the information that a species is found in Malabar province or that a species is usually found in evergreen forests or that a species is found in Karnataka.

   Associations such as the ones described above should be supported efficiently by the data model.

3. There is a need to modify the schema with time in the biodiversity application. For instance, new types of characters may be added to aid the process of identification of a taxon, or new types of geographical organizations may evolve with time. The system should support the facility for easy schema modification without having to recode the application.

3.2 Processing requirements

The modeling requirements focus more on the data types and their relationships. The processing requirements identify the types of queries that need to be answered by the system. The queries on biodiversity databases would primarily be on taxonomy information or geographic information, or on information relating taxonomy and geographic data. The following are some of the major queries in the biodiversity application:

- *Select an order, genera, family or species and list some/all of its characters*
- *Identify a species based on one or more of its identifying characters*
- *List all predators of a given species*

- *Display the map of a geographic location*
- *Display the distribution of climatic conditions in a geographic location*
- *List all species in a given geographic location*
- *List all species with specific characters in a given geographic location*
- *Display the distribution of a species in a geographic location*

We require that the biodiversity information system should be able to efficiently handle queries such as those listed above. For activities such as monitoring the distribution of a species in a geographical region, the system should provide efficient version control schemes.

4 Object Model of OSHADHI

In this section, we propose a biodiversity object model that attempts to satisfy the modeling requirements highlighted in the previous section. This model is implemented in the OSHADHI system.

The development of an object model involves various stages such as identifying the classes, associations, attributes and operations, simplifying the classes using inheritance and finally verifying the existence of access paths for likely queries. We discuss here each of the steps in the object model development. In the following discussion, we use the Rumbaugh notation [Rumb91] for pictorially representing the model.

4.1 Classes

The taxonomy data includes taxons and their characters. The four classes – *Order*, *Family*, *Genera* and *Species* correspond to the four major taxons – the order, the family, the genera and the species. Classes are also required to represent the identifying and non-identifying characters. Therefore, we have the classes *Flower*, *Fruit*, *Seed*, *Leaf*, *Inflorescence*, *Sepal*, *Petal*, *Anther*, *Ovary*, *Style*, *Stigma*, etc., to represent the corresponding identifying characters and a class *OtherChar* to represent non-identifying characters.

In the geographic data model, the classes *Country*, *State*, *District*, *City*, *Forest*, *Biogeographic province*, *Biomes*, *Herbariums* and *Botanical* represent the countries, states, districts, cities, forests, provinces, biomes, herbariums and botanical gardens, respectively. A map associated with a geographic location is modeled as a collection of polygons, lines and points [Scho92]. The classes *Point*, *Line* and *Polygon* are used to represent points, lines and polygons, respectively, and the classes *Points*, *Lines* and *Polygons* correspond to a group of points, a group of lines and a group of polygons, respectively [Scho92]. The *Map* class represents the maps of geographic regions. The *Attribute Temporal Record* class represents the temporal records of an environmental condition in a geographic region and the *Species Temporal Record* class represents the temporal records of a species population in a geographic region. Similarly, the *Attribute Spatial Record* class represents the spatial records of environmental conditions and the *Species Spatial Record* class represents the spatial records of a species distribution in a geographic region.

Maintaining the temporal versions of these spatial records facilitates activities such as monitoring of climatic conditions or species population in any geographic region. The *Value* class corresponds to the values that are associated with these records.

In the bibliographic and the experts data model, the major classes are *Journal*, *Article* and *Author*.

4.2 Aggregations

As discussed in Section 3.1, most of the objects in biodiversity databases are composite, that is, they exhibit an *aggregation* hierarchy. The following are some of the aggregations in the biodiversity databases:

- Taxonomic Aggregation

  An *Order* contains several *Family* objects. A *Family* in turn contains several *Genera* and a *Genera* contains closely related *Species*. Since a family cannot belong to two orders, the order-family relationship is modeled as an aggregation rather than an association. Similar arguments hold for genera and species also. Hence, the taxonomy hierarchy is modeled as a four level containment hierarchy in OSHADHI (Figure 2a).

- Characteristic Aggregation

  A *Flower* character is an aggregation of *Inflorescence* [3] *Sepal*, *Petal*, *Anther*, *Stamen*, *Style* and *Stigma* characters (Figure 2b).

- Geographic Aggregation

  A *Country* contains a number of *State* objects, each *State* is made up of *District* objects and a *District* is made up of *City* objects. This relationship is also modeled as a four level containment hierarchy. Similarly, the association between a *City* and the *Botanical* in the city and the association between a *Country* and a *Forest* are modeled as aggregations (Figure 2c).

- Spatial Aggregation

  The *Spatial Data* also exhibit aggregation. For instance, a *Map* is a collection of *Polygons*, *Lines* and *Points*. In turn, a *Polygon* is composed of *Line* objects and a *Line* is made up of *Point* objects. (Figure 2d).

- Bibliographic Aggregation

  A *Journal* is an aggregation of several *Article* objects (Figure 2e).

4.3 Other Associations

We have seen the various relationships that are modeled as aggregations in OSHADHI. The other associations between object classes are described below:

[3] Inflorescence refers to the arrangement and sequence of development of flowers on a flowering shoot.

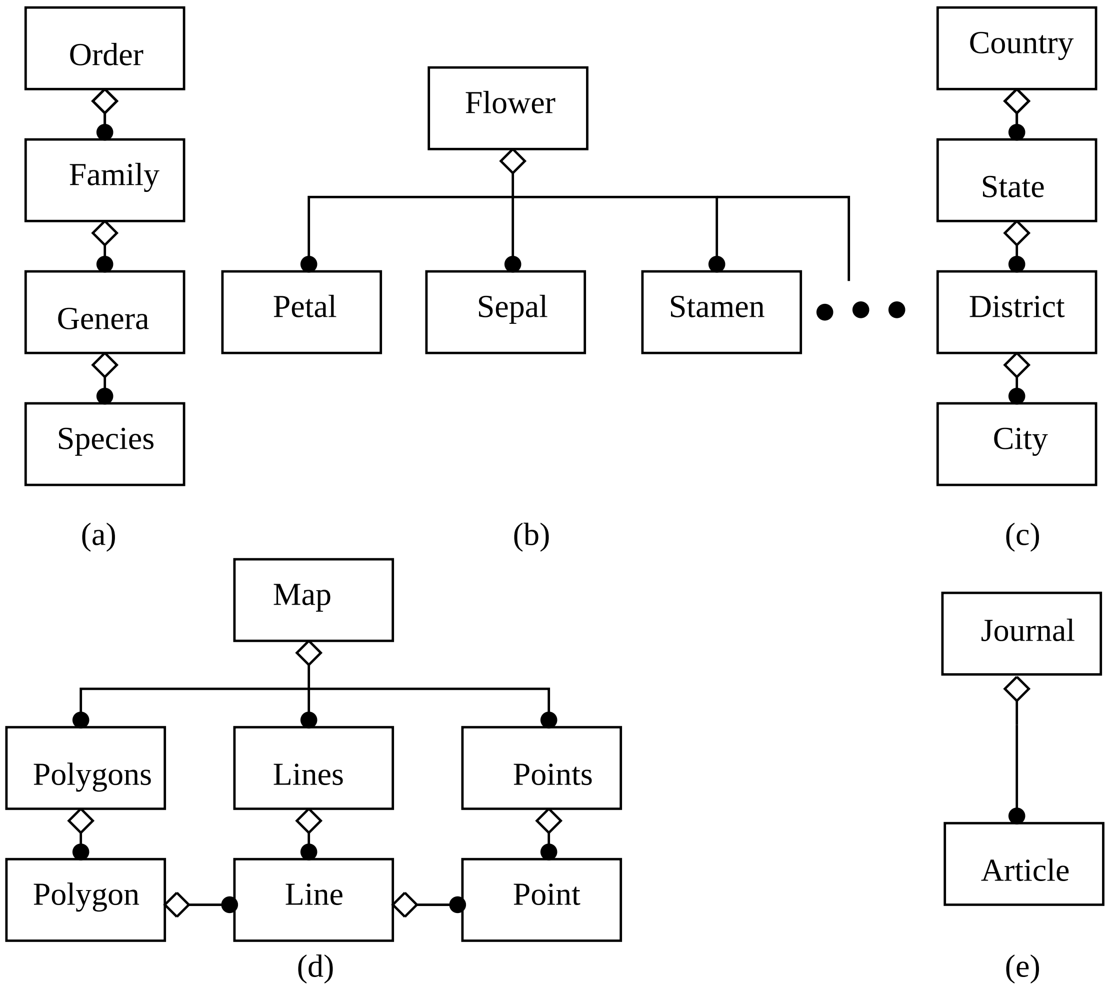

Figure 2: Examples of Aggregations in OSHADHI data model

- Every *Order*, *Family*, *Genera* and *Species* is associated with some identifying characters. Since taxons exhibit a considerable degree of sharing of characters, an identifying character could be associated with many taxons. Therefore, an identifying character cannot be said to *belong* to a specific taxon. Hence, the relationship between the taxons and the identifying characters is modeled as an association rather than an aggregation between the corresponding classes. Modeling the relationship this way helps in the process of identification. This will be seen in Section 7. Every *Order*, *Family*, *Genera* and *Species* is also associated with the *Text* class. The *Text* object models the textual information about any entity (Figure 3a).

- A *Species* is found in many geographic organizations and botanical gardens. This feature is modeled as two associations, one between *Species* and geographic organization, and the other between *Species* and *Botanical Gardens*. A geographic organization or

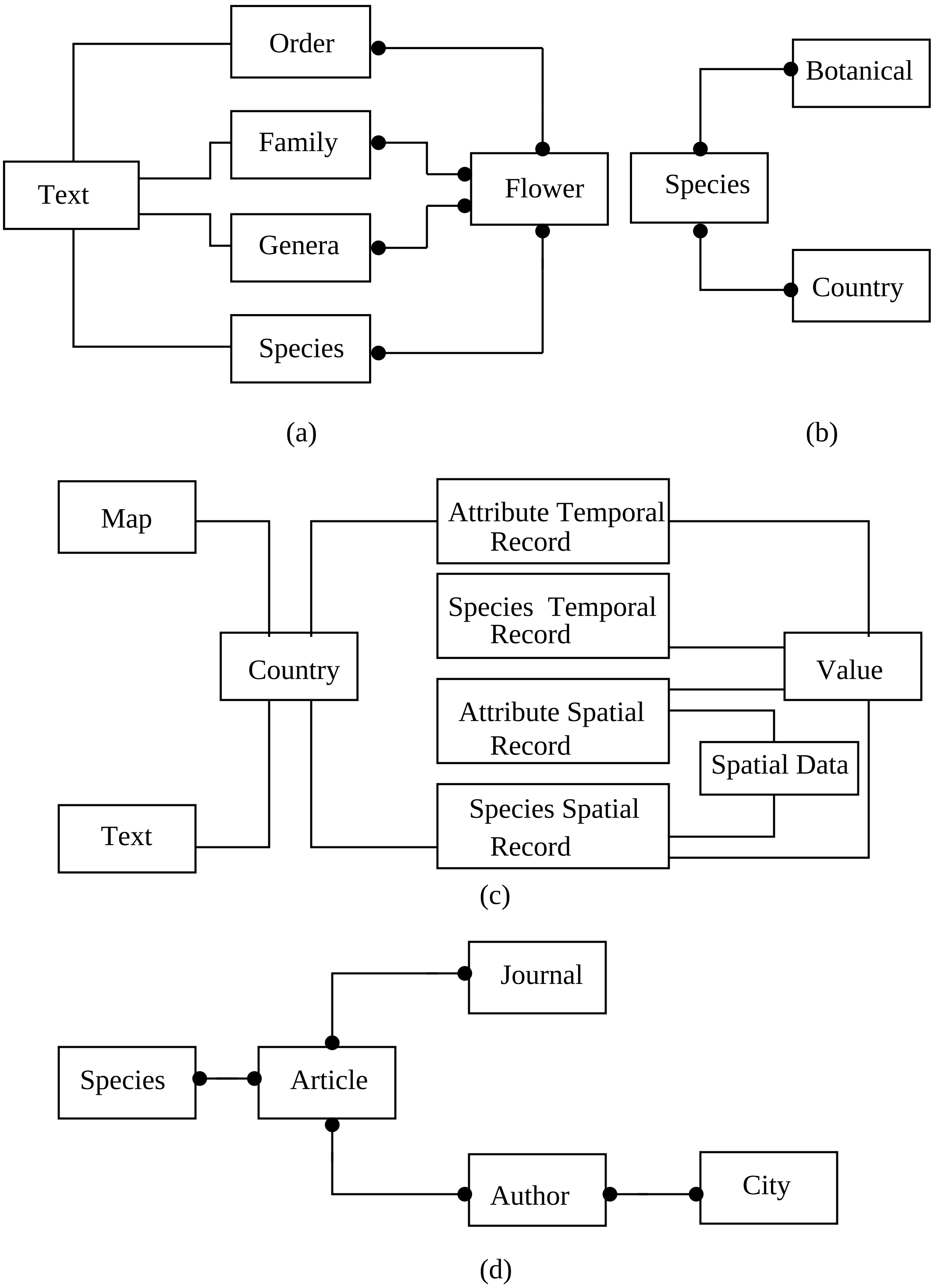

Figure 3: Examples of Associations in OSHADHI data model

botanical garden may have several species associated with it (Figure 3b).

- Geographic regions have *Spatial Data* associated with them. The textual information about a geographic region is modeled as an association between the geographic region and the *Text*. Geographic regions also have several *Temporal Record* and *Spatial Record* objects associated with them. A *Species Spatial Record* is associated with a *Value*, a *Species* and the *Spatial Data* that represent areas. Similarly, an *Attribute Spatial Record* is associated with a *Value*, a name and the *Spatial Data* that represent areas or contours (Figure 3c).

- Information about a species is found in many articles and this is represented by the association between the *Species* class and the *Article* class. An *Article* is associated with its authors, and the *Author* class has an association with a geographical organization (Figure 3d).

4.4 Attributes and Operations of Classes

Attributes are properties of objects that are not objects by themselves [Rumb91]. For example, a taxon has *name* as an attribute; a petal has attributes such as *aestivation* [4] , *color*, *apex*, *form*, and *shape*; and a sepal has attributes such as *symmetry* and *form*.

Most of the methods for the classes are for reading and/or updating the attributes (or objects contained in them, in case of aggregations). The spatial classes such as polygon, line and point have the *Display* method to draw the corresponding object. A polygon has a method to determine if a point is inside the polygon. Similarly, a line has a method to determine if a line intersects it. Methods are also provided to traverse all associations between object classes.

4.5 Inheritance

Some of the classes discussed so far exhibit similarities in their attributes and methods. This gives rise to *generalizations* in the biodiversity data model. The following are the generalizations identified within the biodiversity objects:

- **The Taxon Class and IdentTaxon Class**

  Order, family, genera and species are *Taxons*. The Taxon class generalizes order, family, genera and species. A taxon class has *OtherChar* and *Text* associated with it. Order, Family and Genera have identifying characters. Hence the three are generalized by *IdentTaxon* class (Figure 4a). The classes *Tree*, *Climber* and *Shrub* are specializations (not shown) of *Species*. The *Tree* has additional attributes such as the bark characters. Similarly, *Climber* and *Shrub* have additional attributes.

- **The GeoRegion Class**

  The *GeoRegion* class generalizes country, state, city, bio-geographic province, forest, biome and botanical garden. Every geographical region has a spatial data (map) and a set of records associated with it (Figure 4b).

[4] Aestivation refers to the manner in which petal structures are folded prior to expansion or opening.

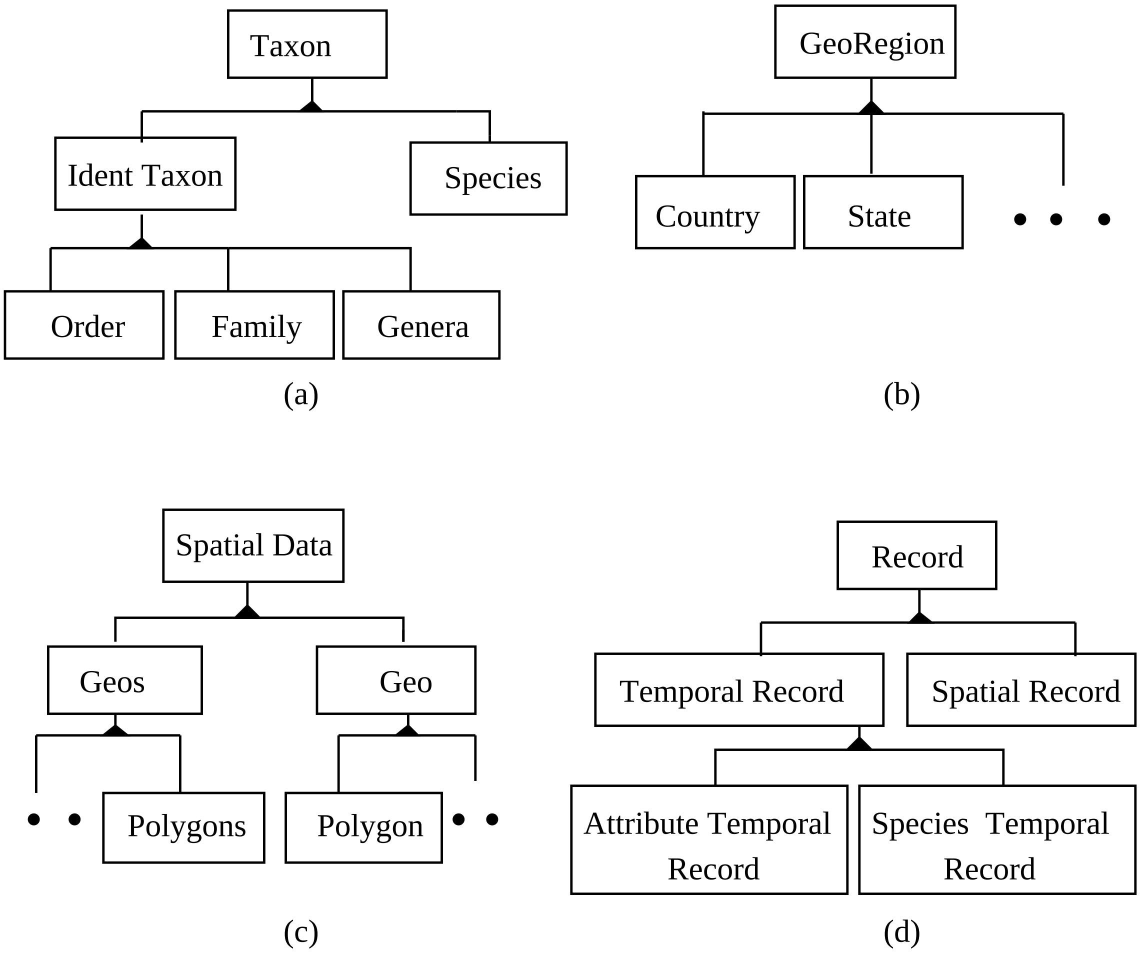

Figure 4: Examples of Inheritance in OSHADHI data model

- **The Spatial Data Class**

  The *Spatial Data* class generalizes all graphical entities. We derive two classes from the *Spatial Data* class : *Geos* and *Geo*. *Geos* is a collection of *Geo*. A *Geo* generalizes the basic graphical entities – *Polygon*, *Line* and *Point*. The *Polygons*, *Lines* and *Points* classes are specializations of *Geos* [Scho92] (Figure 4c).

- **The Record Class**

  The *Temporal Record* class generalizes *Species Temporal Record* and *Attribute Temporal Record*. A *Temporal Record* has a time measure (year, month, date or a time range) and a *Value* associated with it. A *Species Temporal Record* specializes the *Temporal Record* by having an additional association with *Species*. Similarly, an *Attribute Temporal Record* specializes the *Temporal Record* by having an additional property, the *Name* of attribute (e.g., population). A *Spatial Record* generalizes *Species Spatial*

Record and *Attribute Spatial Record* (not shown). A *Spatial Record* has a *Value* and a *Spatial Data* associated with it (since a *Spatial Data* generalizes all graphical entities). A *Species Spatial Record* specializes the *Spatial Record* by having additional association with *Species*. Similarly, an *Attribute Spatial Record* has an additional property, the *Name* of the attribute. A *Record* generalizes *Spatial Record* and *Temporal Record* (Figure 4d).

- **The Value Class**

  The *Value* class forms the super-class for the three primitive classes of *Integer*, *Float* and *String*. A *Range* class, which captures intervals, is derived from the *Value* class.

The proposed object model attempts to satisfy most of the modeling requirements mentioned in the previous section: The *aggregations* are modeled directly in the object model. The abstract class *IdentChar* permits a taxon to be assoc6 mtd with any identifying character. Therefore, the taxon object will contain a collection of object references to the *IdentChar* objects. Similarly, the distribution of a species in a geographic location can be modeled as an association between the species object and the object of type *GeoRegion*. This allows the distribution of a species to be specified at any geographical organization level.

In summary, we see that the aggregation and inheritance features of the OO model cater well to the modeling requirements of the biodiversity application. A complete description of the OSHADHI data model is available in [Vidy95].

5 Architecture of OSHADHI

Having described the data model of OSHADHI, we now move on to describing its architecture, which is shown in Figure 5. OSHADHI is based on a client server model and is built on top of the EXODUS [Care86a, Care86b] extensible object oriented database management system. EXODUS provides the application independent components such as transaction support (e.g., concurrency control, recovery) and storage management and basic version control. EXODUS also provides a persistent programming language called **E**, which is an extension of C++. We have used E to implement the data model and the application specific access methods. Users interact with OSHADHI through a graphical user interface. We now describe the major components of OSHADHI.

- **The EXODUS Storage Manager Server**

  The object storage manager server of EXODUS forms the lowest layer of OSHADHI. The server receives requests from clients for various operations on objects and collections. The server is responsible for transaction support and storage management. The storage manager provides for creation and maintenance of both small objects (objects that do not cross a page boundary) and large objects (objects which extend beyond a page boundary). This is especially necessary in the biodiversity domain where objects mostly tend to be large. The storage manager also provides basic support for versions of storage objects such as the spatial records associated with a geographical organization.

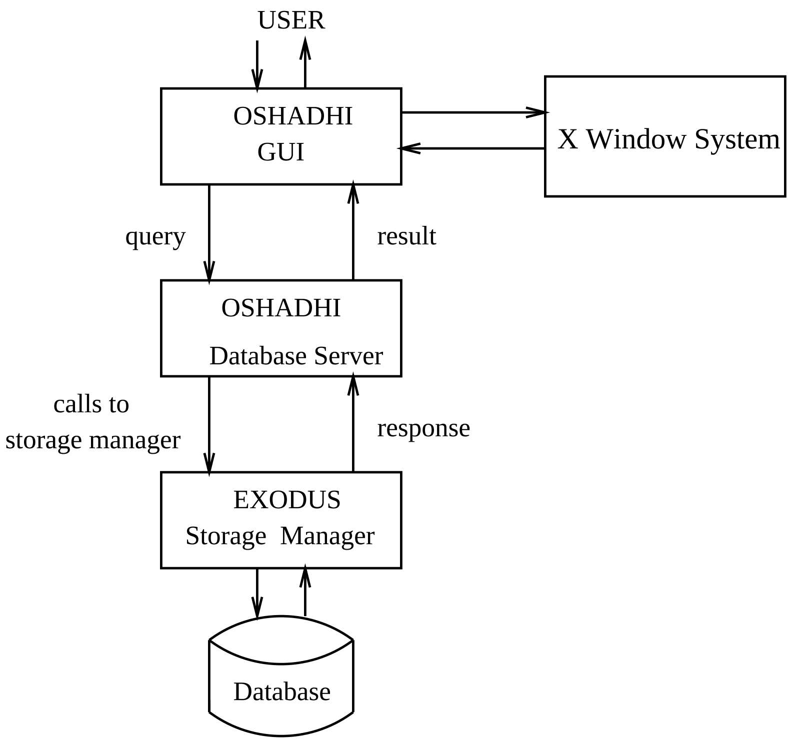

Figure 5: System Structure of OSHADHI

- **The Database Server**

  The database server acts as a client of the EXODUS storage manager server. The database server is responsible for query processing and for all the operations on the biodiversity database. After initialization, the database server waits for a query from its client. The results of the query that are obtained by processing it, are sent back to the client. The database server operates on the database by sending requests to the storage manager of EXODUS through remote procedure calls.

- **The Interface**

  This is a graphical user interface between the user and the database server. The user interactions with the GUI result in a query which is sent to the database server. The interface waits for the result of the query from the database server and displays it in a suitable form when it is received. The GUI is built using X and Motif [Youn90].

6 Access Methods

As mentioned in the Introduction, many biodiversity objects possess certain unique features which may be used to implement query-specific access methods. In this section, we describe the access methods used in OSHADHI. Different types of indexing schemes are used to improve the access time of different types of data objects. Some of the schemes have been built using the libraries already provided by EXODUS while others have been custom designed. We now discuss the various indexes provided in OSHADHI.

6.1 Name Indexes

Objects such as the orders, families, genera, species, countries, states and cities have names. We expect that most of the queries would select these object by name and perform operations on it. We therefore use a B^+tree [Baye72] with the name of the object as the key and this allows retrieval by name. The B^+tree index is provided in the access methods library of EXODUS.

6.2 Character Indexes

The character objects include flower, fruit, seed, inflorescence, etc. Unlike the named objects these objects do not have an attribute which is a primary key on which an index could be built. Searches on the collection of flower or fruit characters could be based on one or more attributes of the objects. Building a B^+tree index on every attribute would be expensive both in terms of space and efficiency, since there are only a few possible values that any attribute could take (typically between 5 and 10). Also, searches involving more than one attribute would need several *intersections* of sets of objects. We therefore use a *numeric encoding* scheme, described below, together with the B^+tree index.

The numeric encoding scheme works as follows: For each character class, the values that each attribute of the class can take are enumerated. Based on this assignment, the numeric value of each of the attributes for an instance of the character class can be obtained. The values of all attributes of the instance are concatenated (the order of concatenation is the same as the order of attributes in the definition of the character class) to yield a number. This number forms a *unique code* for the object and is its primary key. However, the code cannot be directly represented as an integer key, since the concatenation may yield a number that is larger than the maximum integer (usually 2^{32}) that can be represented by the system processor. We therefore obtain a string code corresponding to that number and use it as the primary key for the object. A B^+tree index can then be built with the string code as the key value.

To illustrate the above scheme, consider the character class *SepalChar*. This class represents the sepal character of a taxon. A sepal has the following attributes :

- *Nerve Character* ("not nerved", or "nerved towards apex", or "nerved, but not towards apex")
- *Symmetry Character* ("symmetrical", or "slightly asymmetrical", or "strongly asymmetrical")
- *Form Character* ("oval", or "triangular", or "lanceolate", or "other forms")
- *Fusion Character* ("free", or "fused less than half its length", or "fused greater than half its length", or "fused entire length")

Consider now an instance A of the *SepalChar* class. Suppose this sepal is *nerved towards apex*, *slightly asymmetrical*, *lanceolate* and *free* (for example, as in the species *Alba*). The numeric value assigned to each attribute of the instance corresponds to its position in the list of values that can be taken by the attribute. Therefore the numeric value assigned to attributes of A are 2, 2, 2 and 1, respectively, and the corresponding string code for A would be "2221". We will see how this index is used for searching through the characters database when one or more attributes are specified, in Section 7.

6.3 Quadtree Indexes

In OSHADHI, the user can select a geographical region either by entering the name of the geographical region or by clicking on the appropriate geographical region displayed. When selection is made by name, the B^+tree index is used to access the required object and when the selection is made by clicking on the geographical region, *quadtree indexes* are used to access the required object. Quadtree indexes are built for all geographical regions. Cities (whose spatial data is a point) have a single quadtree index while geographical regions, which are represented by polygons, use two quadtree indexes. The quadtree indexes have been built as described in [Same89], and have been implemented in E.

7 Query Processing

Having seen the various index structures that have been used in OSHADHI, we now move on to presenting the query processing features of OSHADHI. We highlight the major queries and describe how they are processed using the specialized access structures.

Queries in OSHADHI can be classified into three major categories: *Type A:* Queries on taxonomic data, *Type B:* Queries on geographic data, and *Type C:* Queries on taxonomic and geographic data.
We present below an example of each type and describe how it is processed by the OSHADHI server.

1. *Type A :* **Identify taxon with** *sepalnerves nerved-to-apex sepalform lanceolate*

   This query is for identifying a taxon based on one or more of its identifying characters. For example, the above query is for identifying all taxons whose sepals have *lanceolate* form and are *free*. In order to process the query, the character index structure discussed in Section 6 is used. Since the identifying character used here is the *SepalChar*, we use the class description of *SepalChar* given in Section 6. The *SepalChar* has four attributes *nerve*, *symmetry*, *form*, and *fusion*. The *nerve* attribute can take values from 1 through 3, the *symmetry* attribute can take values from 1 through 3, the *form* attribute can take values from 1 through 4 and the *fusion* attribute can take values from 1 through 4. In order to process the query, a sepal *query object* is created as follows: The values of the attributes specified in the query are assigned to the corresponding attributes of the *query object*. For instance, using the class description of *SepalChar* given in Section 6, the *query object* for the above query will have a value 2 (enumerated value for "nerved to apex") for *nerve*, and a value 1 (enumerated value for "lanceolate") for *form*. The unspecified attributes are assigned a value 0 in the *query object*. In this case, the unspecified attributes are the *fusion* and *symmetry*. The code for this *query object* is now obtained as described in Section 6, to be "2020". The required sepal objects are those which have numeric codes of the form "2x2y". In order to obtain these sepal objects, we generate a range query and obtain all the B^+tree nodes whose key values lie between "2020" and "2324" (the 3 and 4 in "2324" correspond to the maximum values that can be taken by the corresponding attributes). However, this set may include many nodes which do not correspond to the "2x2y" pattern (for example, "2030"). Therefore, only nodes whose key is of the form "2x2y" are filtered out from the above set of B^+tree nodes. Using the pointers in this filtered node set, the corresponding sepal objects can be accessed – these objects contain the references to the taxons that possess that character, and these taxons are returned as the result of the query.

The above procedure is repeated for all the required character objects such as inflorescence, petal, leaf, etc. At the end of the process, we have sets of object references to taxons, each set corresponding to a specific identifying character. Each set is a collection of pointers sorted by the name of the taxon to which they point. An *intersection* is performed on the sets of pointers to yield a final set of pointers which point to the required taxons.

2. *Type B* : **Display** *soils* **of country** *India*

   This query requires obtaining the *Attribute Spatial Records* associated with the country *India*, which have a value "Soil" for the attribute *AttrName* and displaying the *Spatial Data* and the *Value* associated with each of them. Other climatic conditions such as *temperature*, *humidity* and *rainfall* may also be viewed similarly. Selection predicates such as "soil is alluvial" may also be specified. We use a B^+tree for indexing the *Attribute Spatial Records*.

3. *Type C* : **List species in state** *Maharashtra* **with** *arrangement polystichou*

   In OSHADHI, users select geographic regions either by entering the name or by clicking on the appropriate geographical object. Quadtree indexes are used to determine the geographical region that contains the point (specified by a *Latitude* and a *Longitude*) that was clicked [Same89].

   The above query requires listing the species in *Maharashtra* which have a *polystichou* floral arrangement. This could be viewed as an extension to identification and is processed by obtaining the pointers to all the species in *Maharashtra* and pointers to all the species which have a *polystichou* floral arrangement (this is done as discussed in Section 6). An *intersection* is performed on the two sets and the result is a set of pointers to the required species.

The other queries supported by OSHADHI include listing various characters and textual information of taxons, viewing the distribution of a species in a geographical region, listing the taxons in the database, etc. The data entry and querying are done through the OSHADHI *Graphical User Interface* which is layered on top of the OSHADHI database server.

8 OSHADHI Graphical User Interface

The OSHADHI GUI serves as the link between the user and the OSHADHI database server. The interface generates the queries from the user interactions and displays the results of the queries obtained from the server in a suitable form.

Data entry is through custom designed forms which have been designed to suit the researchers and scientists at the *Centre for Ecological Sciences* in our institute. The interface is menu driven and queries are generated as the user clicks on appropriate menu buttons. When the user completes data entry, the generated query is sent to the server.

OSHADHI uses simple and user-friendly *form-based* querying that is suitable for the target user group of ecologists. For querying on the geographic database, the user first selects a geographic region of interest. The map of the region is displayed and the options on a side menu allow various queries to be generated. The user may also select a geographic region by clicking on the appropriate region. This highlights the geographic region and queries generated will be on the selected geographic region.

A detailed description of the graphical interface is available in [Vidy95].

9 Conclusions

In this paper, we have addressed a problem which many environmental scientists and biologists face – the requirement of a single source of various kinds of biodiversity related data, and a system to manage this data efficiently and process a variety of queries on them. We identified the specialized modeling and processing requirements of the biodiversity application and proposed an object model for biodiversity databases. We also presented the architecture of OSHADHI, which implements the proposed model.

OSHADHI supports queries such as identification and distribution of species, climatic conditions of geographic regions, etc. and uses specialized index structures such as B^+tree, quadtree, etc., for fast access. Some of the biodiversity data objects exhibit certain unique features which were used to build application-specific access methods. Detailed description of the implementation of OSHADHI is given in [Vidy95].

Several extensions could be made to the OSHADHI prototype. One useful extension would be to include hypertext documents in the database and provide links to objects within the databases. Other extensions such as automatic conversion from the existing DELTA databases are also possible. Although the model has been developed for plant species, we anticipate that the model is easily extensible to animal species also.

Acknowledgements

The work of J. R. Haritsa was supported in part by a research grant from the Dept. of Science and Technology, Govt. of India.

References

[Baye72] Bayer, R., "Symmetric Binary B-Trees, Data Structures and Maintenance Algorithms", *Acta Informatica*, 1972.

[Care86a] Carey, M., et. al., "The EXODUS Extensible DBMS Project : An Overview", Tech. Report, Computer Sciences Dept., Univ. of Wisconsin (Madison), 1986.

[Care86b] Carey, M., et. al., "Object and File Management in the EXODUS Extensible Database System", *Proc. of VLDB Conf.*, August 1986.

[Gutt84] Guttman, A., "R-Trees: A Dynamic Index Structure for Spatial Searching", *Proc. of SIGMOD Conf.*, Boston,MA, May 1984.

[Kort91] Korth, H.F., and Silberschatz, A., *Database System Concepts*, 2nd ed., McGrawHill, 1991.

[Poun95] Pountain, D., "The Poison Disc", *BYTE*, May 1995.

[Rich89] Richardson, J.E., Carey, M.J., and Schuh, D., "The Design of the E Programming Language", Tech. Report, Computer Sciences Dept., Univ. of Wisconsin (Madison), February 1989.

[Rumb91] Rumbaugh, J., et. al., *Object Oriented Modeling and Design*, Prentice Hall, 1991.

[Same89] Samet, H., *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1989.

[Scho92] Scholl, M., and Voisard, A., "Geographic Applications: An Experience with O2", in *Building An Object-Oriented Database System (The Story of O2)*, Banchilon, F., Delobel, C., and Kanellakis, P., eds., Morgan Kaufmann, 1992.

[Vidy95] Vidya, R., "OSHADHI : A Biodiversity Information System", *Master's Thesis*, Dept. of Computer Science and Automation, Indian Inst. of Science, June 1995.

[Youn90] Young. D.A., *The X Window System*, Prentice Hall, 1990.