

DATABASE STRUCTURE OF THE COBOP PROJECT WITH VISUAL INSPECTION VIA WWW

Weilin Hou, Kendall Carder, David Costello, David English, Jim Ivey
Department of Marine Science, University of South Florida
140 7th Ave South, Saint Petersburg, FL 33701

Charles Mazel
Physical Sciences Inc., 20 New England Business Center, Andover MA 01810

ABSTRACT

The Coastal Benthic Optical Properties (CoBOP) program is a 5-year, multi-disciplinary project sponsored by the Office of Naval Research (ONR). Central to the CoBOP project is the study of the optical properties of coastal coral, sea grass, and sediment environments.

More than 20 research groups and 100 instrument systems were involved in a field campaign staged at the Caribbean Marine Research Center on Lee Stocking Island, Bahamas. The database amassed ranged from simple forms (e.g. tables made up of ASCII data) to binary streams, still images, and videos. The various sensor arrays collected physical, chemical and biological data from various sites over a period of two weeks. To better control data quality and enhance access to the data, a web-based query system was designed that allows project participants to visually inspect various data products in graphical form over the World Wide Web (WWW). Due to the complexity of data types, an object-oriented approach was employed to enable researchers to visualize measurements for specified time ranges and geographical locations. Preliminary structure and functionality will be discussed and demonstrated.

BACKGROUND

Proper management and analysis of collected field data can be rather challenging. Considering the difficulties often associated with the field experiment, especially in oceanographic studies, where expensive ship time and research gear has to perform under sometimes rigorous conditions, serious attention must be paid to data management and analysis. When multiple research groups are involved, encompassing multiple disciplines in a multi-year effort, it will not only provide better cooperation among groups, but also help to unearth important links between different aspects of the project. Such is the case for the project Coastal Benthic Optical Properties (CoBOP) program, funded by the Office of Naval Research (ONR). CoBOP is aimed at studying the optical properties of coastal benthic community components such as coral, sea grass and sediment environments, and their overall effects on in-water optical signatures, as well as on water-leaving radiance and remote sensing applications.

With rapid advance in the computer industry, especially the expansion of the Internet and many associated tools and applications, the Internet has become the ideal

candidate to have our goal of data sharing accomplished. There are many issues involved in maintaining data and ensuring fast and easy access. They can be roughly placed into two categories, to reflect the goals we are trying to achieve. First, to ensure fast and accurate sharing of data, a central depository of collected data is desired. These data can be managed by readily available commercial database management systems (DBMS) to ensure flexible access to related data files, although choices are not always straightforward. Second, the ability to look at the content of the data could be very helpful, both in quality control of the data and more importantly, in research. Visual inspection of various data products interactively in graphics from other research groups and across disciplines, and across Internet via WWW, is desired, along with the ability to directly download raw data files, as depicted in Figure 1.

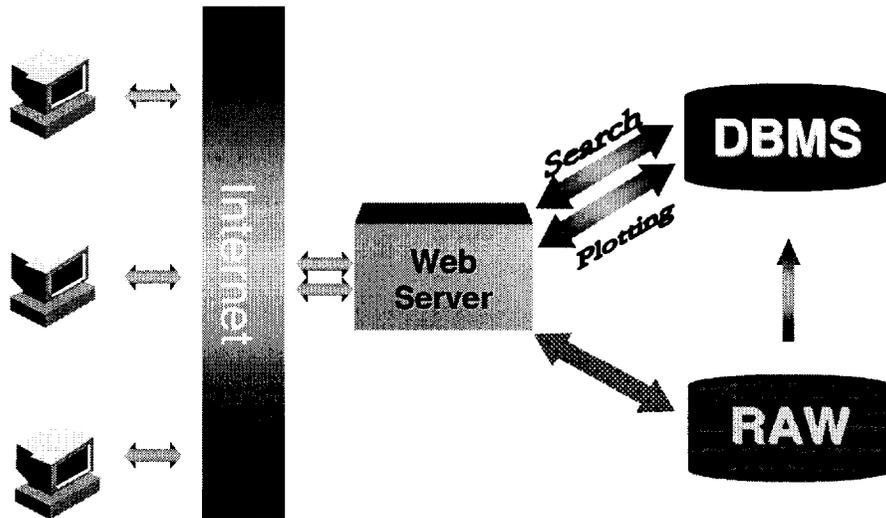


Figure 1. Sketch of CoBOP database project

Data from the recent field experiment near the Caribbean Marine Research Center (CMRC) on Lee Stocking Island, Bahamas, from May 16 to May 30, 1998 are used in this project. Certain instruments, including an airborne image sensor, are not used during this experiment, but will be in the future.

IMPLEMENTATION

An extensive list of instruments are used or will be used in the CoBOP project. They range from in-air and in-water spectroradiometers, spectrophotometers, multi-spectral imagers, to numerous fluorometers and spectrofluorometers, beam transmissometers, scattering meters, CTDs (conductivity, temperature and depth), to current meters, SEM (scanning electron microscope), X-ray diffractometer, stereo cameras and various video and visual inspections (eg. sea grass shoot density) by divers. Depending on experiment configurations and periodic results, the rough number of total instruments used per field campaign is approximately 70 to 100. The number of files generated by each instrument varies greatly, from a dozen per day for most systems to a single file per trip, as with mooring systems, satellite remote sensing coverage, and

overflight of airborne sensors. The size and format of each file type also varies. File size can range from a few kilobytes to tens of megabytes. Their formats range from tabulated ASCII output, to images and video, and other proprietary binary formats. Displaying the contents of these files interactively can be very challenging.

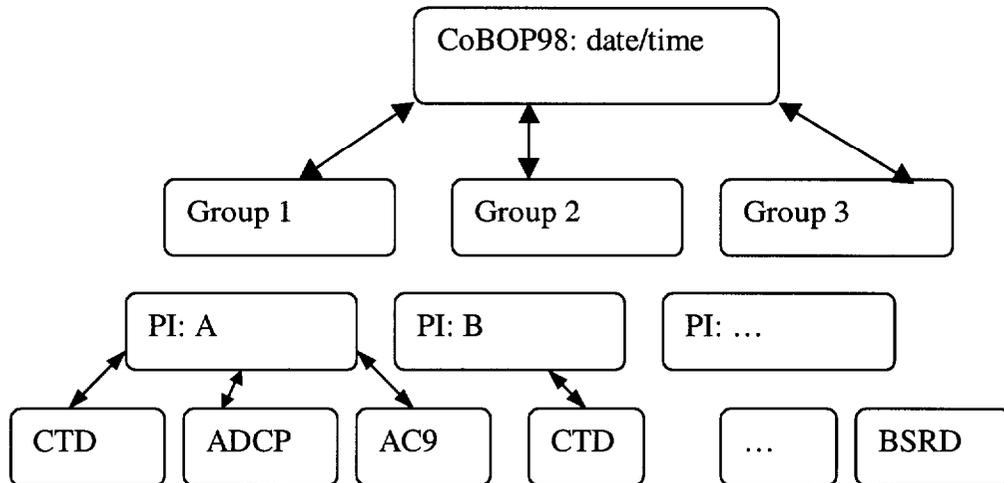


Figure 2. Data structure and class hierarchy in an object-oriented model

In order to access the data over the Internet via WWW, we need to implement local data storage first. A database management system, or DBMS, is designed to store and retrieve information efficiently. Beginning in late 1960s, it used tabulated forms to store information and relations among records, and later retrieve all or selected records by certain criteria, using a standardized Structured Query Language or SQL. These tabulated relational-type databases are currently among the most popular products in the market, such as those from major vendors like Oracle, Sybase and Informix, to list a few. With the advance of computing technology, these database applications are affecting every aspect of our daily lives, such as airline reservation systems and payroll systems.

One major limitation of relational database management system (RDBMS) is that there is a limited number of data types available for each field of the record. Thus, it is not very suitable for some of our peculiar binary types. To overcome such limitation, and more importantly, to better interface with object-oriented programming (OOP) languages such as C++ or Java, an object-oriented database management system (OODBMS) can be implemented, in which everything is an object and you can achieve persistent storage rather easily. Although not widely used, OODBMS is slowly gaining market, such as ObejctStore by Object Design Inc. and Versant from Versant Technologies. One major benefit of OODBMS is that it stores the data structures as they are, without translation to relational or tabulated forms first (Loomis, 1995, Navathe, 1994).

We can see from Figure 2 that this object-oriented approach fits nicely with our data collection model. This hierarchy in data structure can be best modeled by the class hierarchy in OOP, in which sub-class can inherit existing properties from super-class with little or no modifications. The methods defined for super-class can be directly applicable to sub-class objects, or being over-written. This is ideal for our situation.

Suppose both PI A and PI B are using Falmouth Scientific Inc. micro-CTD, but with a slight difference in using their digital channels. We can apply the same processing or plotting function for both instruments with little modification.

To ensure database integrity and automate data input process, we adopt a method that uses a standard description for every data file to be sent in. Therefore, for each data file there will always be a companion file, with the same filename but a different file extension (*.RDM, ensuring no conflict with any data filenames). This is a similar approach to that used in SeaBASS (<http://seabass.gsfc.nasa.gov/~seabass/>). However, we feel that a separate file serves us better, considering that a many of our files are in binary format. The template is also similar to that of SeaBASS. Besides the usual parameters such as beginning and ending time, date, latitude, longitude, Station name, data description, etc; we add other fields such as data type (binary or ASCII), column label, water column depth and bottom type wherever possible. A more detailed description can be found on our web site <http://iceman.marine.usf.edu/cobop/>.

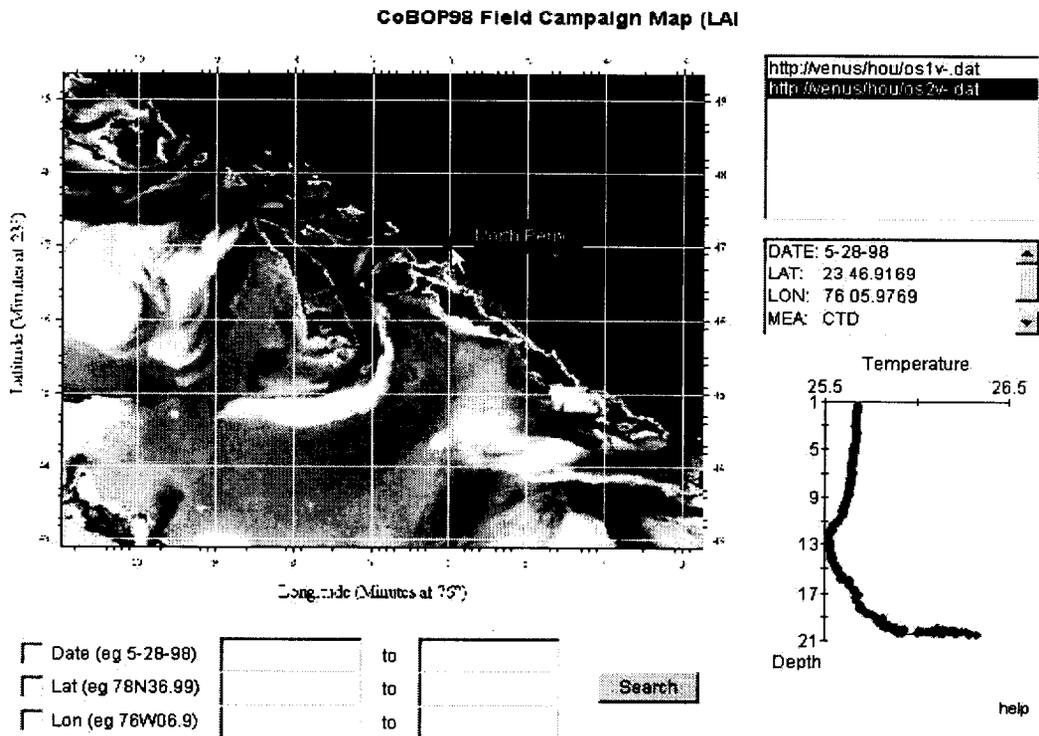
Once the data is stored and can be successfully accessed locally, bringing it over the Internet and adding graphical visualization via interaction is the next step. There are many options to access local data via WWW browser, by means of CGI (common gateway interface), and other proprietary approaches such as RSI's ION server. The option that is the most flexible and still powerful, with smaller overhead is to use the Java programming language. Developed by Sun Microsystems, Java gained instant popularity when it was introduced into the Internet computing world in 1996. Like all OOP, Java can use pre-built components (Beans) to speed up development. The encapsulation of data in the object model can ensure security over the network. Instead of producing all results on the host or server machine and then transmitting them over the network, Java enables users to bring in the code and related data to the local machine to compile then to execute in a universal environment, or Java Virtual Machine (Flanagan, 1997). This approach trades local compiling time for network traffic and is very suitable for Internet computing, as it also implements the goal of cross-platform compatibility -- "write once, run anywhere".

Another issue to consider during development is the cost. We currently utilize Microsoft Windows NT as our development and deployment platform, for reasons of lower cost as well as abundance of various development environments. To shorten development time, we used a somewhat hybrid approach, storing data descriptions in RDBMS format, and defining individual data files as objects when plotting is desired. This puts limits on interactive plotting between different files. Microsoft SQL Server 6.5 is used along with Symantec dbANYWHERE to communicate with applets built via Symantec Visual Café for Java Database Development IDE (Integrated Development Environment).

DEMONSTRATION

As discussed above, we need an interface that takes advantage of the Java programming language for faster transmission and a better development environment. At the same time, we need to deal with the interface with database functions so that we can query availability of data and then attempt to display the content interactively.

These functions are partially illustrated in Figure 3, a screen capture of the applet in action with only a few data files loaded. The applet consists of a main image map, where stations can be displayed when the mouse pointer moves in its vicinity (as “North Perry” in Figure 3). When this label is enabled the linked files are displayed in the upper right corner box (for convenience, let’s call it FILE box) . When you double-click on one of the items in this box the content of the file is displayed in a box just below it (indicated by “DATE: 5-28-98”...), along with a plot of the file content. This plot is zoom-able. You can click on any part of the plot to zoom in, or drag and draw a box of interested region to display details. There is a simple “help” screen that will pop up over this plot region once your mouse pointer moves over the little blue “help” region located in the lower right corner. Once you move away from “help”, your original plotting will resume.



Applet started.

Figure 3. Screen capture of demonstration of database project via WWW

The text boxes below the main image map are designed to do some simple searching against the database connected to this applet. The search result is displayed in the same FILE box mentioned earlier. Other operations applicable to the FILE box can be performed from then on. Above the image map there is a scrolling bar to notify of upcoming events, and it can also be used to update information about changes made to this applet.

It is worth mentioning that we do not have all the database connections worked out at this stage. The relational approach to store information about the data files prevents loading original data into the system. Codings are done to link plotting with associated limited type of files at this stage. Due to different event models and security, Java

codings conform to JDK 1.1.5 (Java Development Kit), therefore you will need to use version of Netscape no later than 4.04 (with JDK 1.1 patch) or 4.06, and Microsoft Internet Explorer 4.0 SP1.

DISCUSSION

Our goal is to have a final product that is easy, fast and efficient. It should be easy to use, with a simple graphical user interface (GUI), while at the same time, it should be easy to develop. The Java development environment can meet such criteria. In order to use over the Internet efficiently, the less overhead the better. Approaches that generate local image files of plotting results and then transmit over the network by means CGI script should only be considered when other programming cannot meet the need. Besides the transmission time, another factor limiting the performance is how well the database engine operates, especially when the database is huge with complex relations. Research shows that in these situations an object-oriented database should be considered².

An alternative to the above approach is to use geographical information system (GIS), systems such as ESRI's ARCView, to overlay a great deal of geographical information on an image map. Their Internet Map Server (IMS) then allows access to such information over the network. However, due to lack of structural information, coding interactive plotting with Java to work together with the GIS can be a challenging task.

The best approach to this database project is strictly object-oriented. It can deal with original data with ease. Interactive plotting against similar measurements can be performed with little coding and makes it possible to compare different sources of data. However, due to current lack of popularity and thus lack of development tools support, longer development cycle can be expected. Although it has been touted as the next generation of database management systems since the early 1990s, it has until now gained limited success. Mixed models, or hybrid models between relational and object-oriented architectures are emerging, such as recent release of Oracle 8. Thus, it is promising that soon we could have the versatility of object database with richness of development tools in RDBMS.

ACKNOWLEDGEMENT

Financial support was provided by the Office of Naval Research to the University of South Florida through Grant N00014-97-1-0006 and N00014-96-1-5013. We also thank CMRC for their help.

REFERENCES

- Flanagan, David, Java in a Nutshell, 2nd Ed., O'Reilly, 1997
- Loomis, Mary E. S., Object Databases: The Essentials, Addison Wesley, 1995
- Navathe, Shamkant B. and Elmasri, Ramez A., Fundamentals of Database Systems, Addison Wesley, 1994