	gb/doc/gb.inst.ps (PostScipt format)
dm.tar.Z	DM database manager and view-server
ascii.tar.Z	ASCII database manager and view-server
sdm.tar.Z	SDM database manager and view-server, and rshowview, rviewsdm, rassocviews commands
sql.tar.Z	SYBASE and UNIFY sql-view-servers
parse.tar.Z	mathematical expression parser subroutines
lxt.tar.Z	Lamont X-Toolkit subroutine library

Table 3. Public view-servers at the Lamont-Doherty Geological Observatory.

Name	Number	Type	Contents
miles.ldgo.columbia.edu	129.236.10.70	DM	misc. geophysical
duke.ldgo.columbia.edu	129.236.10.50	SYBASE	NCEER strong motion
trane.ldgo.columbia.edu	129.236.10.90	SDM	misc. seismic
chaos.ldgo.columbia.edu	129.236.10.20	ASCII	misc. geophysical
clipper.ldgo.columbia.edu	129.236.20.90	GMT	marine geophysical ship data
ocean.ldgo.columbia.edu	129.236.20.71	GMT	marine geophysical ship data

Fig. 1. Data accessing system. The scientist's computer at Site 1, which is running a map based geographical database browsing program (e.g. GB) can retrieve data from data serving computers at Site 2 and Site 3, located elsewhere on Internet.

Fig. 2. Screen of scientist's workstation during typical use of the Lamont Geographical Database Browser (GB). The scientist is displaying a map of western North America. Coast lines and 1000 meter topographic profiles are shown for orientation. Earthquake hypocenters from the NCEER_-EVENT view (crosses) have been retrieved from the view-server on the computer duke.ldgo.columbia.edu, which is maintained by the National Center for Earthquake Engineering Research. The scientist has selected one of these events (large cross) and retrieved records from the NCEER_-TRACE view for that earthquake. The scientists has selected one record from this view, corresponding to a seismic station located near the hypocenter (square). One of the fields of this view contains the pathname of the strong motion accelerogram recorded at that station. This time series is displayed using a separate program.

Table 1. Sample 'earthquake_hypocenter' view. Note that the data is organized into columns (fields) and rows (records). Each field has a name and a data type.

lat	lon	source	epoch_time	depth	magnitude
DOUBLE	DOUBLE	STR 6	DOUBLE	FLOAT	FLOAT
70.891	52.868	GS	120558080.7	0	4.4
70.7	53.5	CGS	-258470951.2	0	0
70.7	53.5	CGS	-258132573.4	0	0
70.756	53.872	GS	117961198	0	6
70.807	53.231	GS	120557036.3	0	4.2
70.682	53.545	GS	143688741.5	0	4.4
70.843	53.69	GS	182854796.3	0	6.7
70.779	54.177	GS	120553197.4	0	6.9
70.817	54.063	GS	152600396.7	0	6.7
71.297	51.875	GS	120561231.3	0	4.8
71	52.5	CGS	-385837516	0	0
72.9	55.2	CGS	-166809605.2	0	4.3
72.914	55.858	GS	523288596.65	10	4.7
73.7	53.8	CGS	-260123345.2	0	0
73.7	55	CGS	-229345048.3	0	0
73.315	55.146	CGS	24731997.1	0	6.7
73.387	55.1	ERL	54799195.2	0	6.4
73.336	55.085	ERL	83829596.5	0	6.3
73.302	55.161	GS	116665194.3	0	6.8
73.366	55.094	GS	147002395.5	0	6.4
73.351	55.078	GS	183124797.3	0	6.5
73.6	56.2	CGS	-257787042.8	0	0
73.4	56.5	CGS	-221480642.8	0	0
73.6	57.5	CGS	-221575698	0	0

Table 2. Compressed UNIX TAR files and their contents.

name	contents
gb.lic.ps	License Agreement (PostScript Format)
gb.tar.Z	GB geographical browser, and manual for entire system,

sions operating on the fields of the views. The user can call up any number of views, and plot them on a base map (either as discrete symbols for parameters, or by contouring images). GB can also be used to transfer data to other programs, including spreadsheets, time-series analysis programs and 2- and 3-dimensional plotting programs, all of which use common data formats.

AVAILABILITY OF SOFTWARE AND ACCESS TO VIEW-SERVERS

We encourage interested scientists and data management staff to consider running view-servers on their institution's computers. Such an effort will be particularly easy if you have access to Sun workstations, since recompilation of the software will not be necessary. First transfer the manual, gb.inst.ps, and the rshowviews, rviewsdm, rassocviews programs to your computer (see Table 2). You will then be able to access view information and data from view-servers (see Table 3). Then transfer and install GB, which retrieves data and displays them on maps. Finally, pick one of the simpler view-servers (e.g. ASCII or DM), install it on your computer, and build some of your own databases. Unfortunately, due to our limited resources, we cannot offer you any technical support in this endeavour.

License to use the software described in this article is granted free of charge to non-profit educational institutions and U.S. Government Agencies, by simply completing and returning the License Agreement in the document gb.lic.ps (see Table 2). All material can be obtained by anonymous FTP to lamont.ldgo.columbia.edu (129.236.10.30) and consists of compressed UNIX 'tar' files in the directory ~ftp/pub (see Table 2). In return for the use of this software, we request that your host institutions be diligent in maintaining databases and making them publicly available. We provide a directory service for view-servers through the view named 'view_servers' on chaos.ldgo.columbia.edu. Some interesting time series and raster data are available on the public NFS-mountable file system duke.ldgo.columbia.edu:/duke/nceer_data. E-Mail should be addressed to menke@lamont.ldgo.columbia.edu to announce the existence of view-servers.

Acknowledgments. We thank all the people who over the years contributed to Lamont's software development efforts, including members of our programming staff: Richard Boaz, Roger Davis, Ken Howard, Yalin Liu, Brenda Murphy, Suzanne O'Hara, Doug Shearer; engineers: Dale Chayes, David Lentrichia, Larry Shengold and scientists: Dallas Abbott, Tom Boyd, Alberto Malinverno, Walter Smith, Jeffrey Weissel, Paul Wessel, and Dean Witte. We also thank people from other institutions for their comments and assistance, including Bill Leith and Janet Walz of the U.S. Geological Survey. Parts of this research was funded by the National Science Foundation under grants _____ and _____. Lamont-Doherty Contribution Number 00000.

few days of modest programming effort. Special purpose view-servers can be written to provide a network interface even to one-of-the kind databases. For instance, Lamont has a view-server for its 'GMT' marine geophysical database.

SIMPLE COMMAND-LINE ACCESS TO VIEW-SERVERS

A 'user-interface' is a program that provides a scientist with access to data made available by the view-servers. The simplest user interface is a set of command-line oriented programs that parallel the commands of LVS protocol. For instance, information about the views made available by the view-server running on a given computer can be obtained with the command:

```
rshowviews hostname
```

where 'hostname' is the Internet name of the computer running the view-server. Once the scientists is aware of an interesting view, data from that view for a particular part of the world can be obtained with the command:

```
rviewsdm host view_name query left right bottom top
```

Here 'query' is a logical expression (in a C-like syntax) operating on the fields of a single record of the view, and 'left', 'right', 'bottom' and 'top' are the longitude and latitude bounds of a rectangular region on the earth. For instance, if the view 'ISC_hypocenters' contained a field 'magnitude', the scientist could recover all hypocenters from a given region with the query 'magnitude>5.5'. The data retrieved by this command are in a simple, ASCII 'spreadsheet' format.

The scientist then can inquire whether there are any other views related to a particular view with the command:

```
rassocviews host view_name
```

Perhaps, following the example above, the 'ISC_hypocenters' view is associated with a 'ISC_arrival_time' view. Then the scientist could go on to recover the arrival times associated with a particular earthquake with the command:

```
rviewsdm host view_name query target_id
```

Here the 'target_id' would be the ID of the hypocenter of interest.

HIGH PERFORMANCE MAP-BASED USER INTERFACE

The Lamont Geographical Database Browser (GB), is an X-Windows (and SunView) based GIS application which can display data in the form of symbols on maps (Figure 2). This application learns from network view-servers about what types of data are currently available anywhere on the Internet and determines their schema. The user can make queries to retrieve these data and can manipulate them graphically and symbolically. This software employs a powerful mathematical expression interpreter that allows the user to make queries from the database, perform sorting and arithmetic functions, and control symbol attributes, all through symbolic expres-

standard language for manipulating databases, could be used directly. We decided it could not, for three reasons: such a choice would seem to preclude the use of fast, simple non-SQL databases which are already in use in many university data archives; in actual practice, relational database managers from different vendors use substantially incompatible versions of SQL and completely different methods of communicating SQL queries to database managers; and finally, SQL lacks standard features for documenting the contents of databases. In our opinion, the ability to recover information *about* the data in a standard way is every bit as important as the ability to recover the data itself.

EXISTING VIEW-SERVERS AND UNDERLYING DBM'S

We have designed and implemented four view-servers, each which uses a different database management system:

1. The ascii-view-server was designed to quickly get data into a database and begin to examine and manipulate it. Hence the ascii-view-server accepts text files in a simple, 'spreadsheet' format, with each spreadsheet defining its own view. For instance, a database containing chemical analyses of rocks, originally stored in a commercial spreadsheet format, can be made accessible with a few minutes of effort.

2. The dm-view-server, and associated 'DM' database manager was designed to allow very fast access to large, flat (i.e. single table) databases. Data can be quickly entered into DM databases from spreadsheets. This database manager has proved suitable for large (e.g. 10-100 Mbyte) tables of earthquake hypocenters, satellite track data, etc.

3. The sdm-view-server, and associated 'SDM' database manager was designed as a simple extension of DM. It manages a hierarchical arrangement of tables (e.g. inverted tree). This software provides much of the functionality of commercial database packages, but, along with the ASCII and DM view-servers, can be freely distributed.

4. The sql-view-server provides an interface between LVS protocol and relational database managers (RDBM's) that use Structured Query Language (SQL). The view-server's initialization file contains instructions and SQL fragments that indicate how to use the user's request for data to formulate an SQL statement. This statement is then sent to the RDBM's SQL program, and the reply is converted back to LVS protocol. Since SQL requests and replies consist of ASCII text, the sql-view-server is fairly inefficient in recovering data from the database. On the other hand, the use of SQL allows many complicated tables to be easily joined in a single view. Unfortunately, SQL is still sufficiently non-standard that we have several versions of the sql-view-server, each optimized for a different vendor's SQL (e.g. UNIFY and SYBASE). Ironically, although commercial database managers are generally much more flexible, robust, and powerful than university-written ones, our experience is that they require more set-up time, with more steps required to build a database. We have attempted to speed the database building process by implementing a program that generates a data base creating SQL script from simple, spreadsheet input.

We have designed LVS protocol to be simple enough that a view-server can be implemented in a

5. list of *field_types*. Fields must be of the following primitive data types: short integer, long integer, single precision floating point, double precision floating point and character string. Character strings are limited to 255 characters in length.

6. list of *field_lengths*, in bytes.

7. list of *field_comments*. A field comment describes the contents of a field, and gives it units.

8. *coarse_index*, with one bit for each of the 64800 degree squares on the earth (i.e. 8100 bytes). A bit is set if the view has data for the corresponding degree square. Since the distribution of data may be difficult for some DBM's to determine, a coarse index which is completely set is acceptable. Some views may not have an associated (lat, lon) in which case the coarse index is meaningless and a coarse index which is completely empty may be given.

9. *view description*, a paragraph of text describing the view, noting updates, relevant publications, and authors.

The network protocol used by the view-servers (called 'LVS protocol') is implemented in architecture-independent External Data Representation (XDR) format using Berkeley sockets as the underlying communication method. LVS protocol implements only four basic commands:

1. Get all schema: get all the schema (and other descriptive information) of all the views it is aware of.

2. Get a specific schema: given a view name, get its schema and other descriptive information.

3. Get related views: given the name of a particular view, say view_a, suggest names of other views, say view_b and view_c, where a record from view_a implies one or more records from view_b and view_c.

4. Get data: Get data for a view, given a query. Access may use one of two broad methods, geographical (which gets all the records of a view in a given geographical region) or by identifier (ID), which gets all the records that have a specific ID. The ID method of access is usually used in connection with the logical relations of views (described in 3, above), that is, the user has a record of view_a, and now wants to get the implied records of view_b or view_c. The ID of a view may be a short integer, long integer, character string, or the complete record of another view. A query consists of a geographical region or list of ID's, and a logical expression that each of the records must pass.

Before designing the LVS protocol, we considered whether Structured Query Language (SQL), a

second on a fast workstation (e.g. Sun Sparkstation 1), so that the access time is limited mainly by the ability of the database manager to extract data from the database. For instance, 'UNIFY' (a commercial relational DBM) typically extracts data at a rate of 20 records per second, while Lamont's 'DM' flat table DBM extracts data at about 800 records per second. All data are sent over the network from the view-server to the user-interface in External Data Representation (XDR), so that problems do not arise with computers that use different byte ordering, etc.

C. User Friendliness: to provide a well-documented way of novice access to complicated databases. A large database with a complicated schema can be very intimidating to novice users. The problem is that there are often too many tables for someone to figure out how to join them together to produce useful results. Furthermore, many database managers fail to provide a way to retrieve an explanation of the database (a general description of its comments and its schema). The view-server is designed to provide this level of information about the database, even when the underlying database manager cannot.

D. Robustness: to provide a stateless network service that will not cause the database serving computer to fail when one of its clients fails in some way. A separate view-server process is created (forked) for each view-server command. The process - but not the computer - will die if the database serving computer loses contact with a client or if the client issues nonsensical commands. Other programs running on the database serving computer - including communication with other view-server clients - will be unaffected.

At the heart of the view-server concept is the 'view' itself. The user requests records from a 'view' of the database. A view is a rectangular table of rows (or 'records') and columns (or 'fields') constructed from the tables of the database by the view-server. The user has no control over the definition of a view - only the scientist archiving the data does. But a user can determine which views are currently available, and - most importantly - can determine their schema (a set of information that completely describes the view). The schema of a view consists of:

1. *view_name*, a character string, such as 'ISC_hypocenters'.

2. *view_method*. A view is either accessed by geographical location or by an identifier (which may or may not be unique).

3. *number_of_fields*, the number of fields in the view.

4. list of *field_names*. By convention, certain field names have special, reserved, meanings. For instance, 'lat' and 'lon' are reserved for latitude and longitude in decimal degrees north of the equator and east of Greenwich, and 'id' is reserved for an identification number (ID). Connectivity of points, as in a ship or satellite track, is maintained through a 'successor_id' field (the ID of the next point in the track).

can get views of data from the database (Figure 1). Each view is a simple, rectangular table of data (Table 1). The structure of a view is intentionally kept simple so that it is easy to write software that can learn the schema of new views from a view-server and present the data in a view in some desirable form (e.g. as graphs, tables, or maps) to the scientists. The ability of the scientist's display software to manipulate a particular view is not built in, but rather is learned from the view-server. This automatic learning facility improves the software's versatility.

In order to further simplify the design of applications that can communicate with the view-server, we have limited the fields of views to parametric data (integers, floating point numbers, and character strings). We handle times series and images in a rather ad hoc fashion, by assuming that they are stored in a pre-defined format in a file, and including the file pathname as a field of the view. Standard Internet file transport (e.g. FTP or NFS) is then used to recover the time series or image.

Our main effort in application development has been a geographical database browser (called 'GB'). This application is the main tool for extracting data from databases. It also has most of the features of a traditional GIS application: a scientist can mathematically manipulate parametric data and images and display them on maps. GB can be used in conjunction with other software (including commercial GIS applications), as part of a larger research environment. A key part of the overall philosophy of the Lamont system is to keep the different parts of the system as loosely coupled as possible, with just enough standards to maintain a 'look and feel' similarity between applications and to facilitate interchange of data between them. Thus the system can easily be built upon by specialized users. For example, Lamont's Seismology division has developed several substantial seismogram manipulation and earthquake location applications which are on the one hand completely distinct from the data access system, and on the other hand function smoothly together with it**.**

VIEW-SERVERS FOR PUBLIC ACCESS OF PARAMETRIC DATA

A view-server provides a uniform, network-based access method for publicly accessible geographical databases. The problems that it tries to solve are:

A. Masking: to hide the details of a particular database manager from the user interface. The user interface should mainly be concerned with displaying and manipulating the data, not with supporting a myriad of underlying database managers. In our model, many view-servers may be running on different computers connected by Internet, one view-server per computer. Usually a single view-server provides access to a family of related databases all created by the same DBM software. Each view-server fulfills requests from users elsewhere on Internet, without regard for the actual location of these users. All view-servers talk a standard language (or 'protocol') with the user interface, regardless of the underlying database manager. The view-server gives the user interface limited read access to the database, presenting only the schema and data of the selected views (virtual flat tables built from the tables of the database). There need not be a one-to-one correspondence between views and tables.

B. Platform Independence: to provide fast, architecture-independent retrieval of data. The view-server can process records at a rate in excess of 1000 records per

is becoming worse. Second, the data must be readily available (preferably on-line). A delay of a month while a scientist attempts to obtain a tape of a critical data set would greatly interfere with a study, and especially with pursuing promising (but speculative) paths during such a study. The available data should also be complete and up to date, which is to say that they should be obtained as close as possible to their original source, as contrasted to using incomplete copies of unknown pedigree. Third, the method of data access needs to be reasonably uniform, even though the individual data sets may be stored in physically different places and by different methods. A scientist should not have to deal with a dozen or so idiosyncratic data access methods in order to access a dozen kinds of data. Fourth, there is a critical need for user-friendly software applications that can access, display (e.g. in map format) and manipulate data. Finally, the whole system must be flexible enough that it allows for scientific creativity in an exploratory research environment. The various pieces of the system must be coupled sufficiently loosely that they can be replaced with new, experimental, or custom made, pieces. The system must not be self-contained, but instead only one facet of an overall research environment.

No system which comes close to embodying these features is currently available (though many data management and display systems contain clever, innovative ideas worth emulating). Most of the commercial Geographical Information Systems (GIS) are either very limited in their capabilities (to, say, satellite photographs) or very rigidly structured, in a way that might be suitable for a production, but not a research, environment. Furthermore, most existing GIS software provide only minimal data management and data access services. We at Lamont therefore developed a prototype system in order to work out some of the theoretical and practical problems involved. Development of these tools has a high potential of leading to scientific advances, since it simultaneously frees the scientist of the drudgery of assembling a data set, and provides a larger set of data from which new insights can be gained.

DESIGN OF A DATA ACCESS SYSTEM

Our design was primarily motivated by the way in which Lamont (and many other labs) function, with small groups of scientists collecting and archiving a given type of data, but with a much wider community of scientists accessing (reading) that data. The scientists who collect the data want primary control over the data. They want the data to reside on a nearby computer system. They want to choose their own data management system, design their own database structure, and control changes made to the database. Since they are the experts on that data, they need the most sophisticated access to the data. Other scientists have only read access to the data, and may be willing to trade off flexibility for convenience of access. For instance, if the data are stored under a relational database manager, they may be satisfied with a few predefined ‘views’ (where a view is a single flat table built from the many tables of the database) rather than general access to the complex database itself.

We therefore decided to distinguish between these two types of access to the data. We assume that the data are stored using some database management (DBM) software, which may be of university or commercial origin. This DBM software is used for inputting data into databases, and can be invoked directly by the scientist for ‘expert’ access to the data. We then designed an intermediate layer of software, a ‘view-server’, that separates and insulates the ordinary scientist from complexity of the DBM software and of the database itself. The view-server keeps track of which databases are available (there may be many, each controlled by a different small group of scientists, each at a different location on the local - or wide - area network), and their schema, and

A Voluntary, Public Method for Sharing Earth Science Data over the Internet using the Lamont View-Server System

William Menke*, Paul Friberg, Arthur Lerner-Lam, David Simpson,
Robert Bookbinder and Garry Karner
Lamont-Doherty Geological Observatory of Columbia University
*also Dept. of Geological Science, Columbia University
* Internet E-Mail: menke@ldgo.columbia.edu

for submission to EOS, March 8, 1991

Abstract.We envision a time when an earth scientist from any institution will be able to access high quality data, regardless of where this data may actually be archived. The scientist will not have to know anything about where the data actually resides or what format it is stored in. He or she will only have to specify the type of data and the workstation software will handle the rest of the details of finding them and accessing them. A method - now in use at Lamont and several other institutions - of achieving this vision is described in this paper. Institutions make 'views' of their databases publicly available to users of the Internet, using database serving software that runs on one of their computers. This software completely automates the process of finding out what kind of data are available and of retrieving them. Hence a wide variety of different databases become, from the scientist's perspective, parts of an Internet-wide earth science database. At the same time, the institution that archives the data is not locked into any particular database management system, and is free to provide alternative access methods and to exclude some of its data from the system. The self-teaching nature of this concept is especially useful in large cooperative scientific efforts.

INTRODUCTION

In order to make scientific advances in understanding a given region of the earth, an earth scientist may need to study and compare many different data types, having very different structures and data volumes. For instance, in a geological study of a volcanic region, a scientist may need to examine topographic data in order to locate the major volcanic edifices, gravity anomaly data in order to make inferences about crustal thickness, seismicity data in order to locate active geologic faults and regions of magma movement, satellite photos in order to map recent lava flows, and chemical analyses of rocks in order to understand the magma generation process. Furthermore, a large number of subsidiary databases (such as magnetic anomaly, heat flow measurements, well temperatures, meteorological reports, etc.) may play a role in resolving a particular problem.

Research in such integrated studies can be greatly facilitated by a carefully designed database tool. Such a tool must possess certain properties. First, it must be capable of teaching the user about what data are available. Too often in the past a researcher has overlooked a source of data that might further his or her research out of ignorance of its existence (even though a nearby colleague may have it in a desk drawer) or because the structure of the database was perceived as being too complex. As time progresses, more and more data are being collected, in much larger volumes than previously, by a larger and more international group of people, so that this problem