The United Nations University

GEOTHERMAL TRAINING PROGRAMME
Orkustofnun, Grensásvegur 9,
IS-108 Reykjavík, Iceland

Reports 1997
Number 16

A PRELIMINARY DATABASE DESIGN FOR GEOTHERMAL FIELDS IN EL SALVADOR

Ricardo Alberto Ventura Hurtado
Comisión Ejecutiva Hidroeléctrica del Río Lempa (CEL),
Gerencia de Investigaciones Geotérmicas,
km 11 ½ Carretera al Puerto La Libertad,
Santa Tecla, La Libertad,
EL SALVADOR, C.A.

ABSTRACT

A preliminary database design has been carried out for two El Salvador geothermal fields. The database contains thermodynamic information, such as downhole temperatures and pressures, wellhead mass and heat flow with time. The database is considered a necessary component in modern computer networks, consisting of central and peripherical devices, where communication takes place between the clients and servers and different databases. In this work an Oracle relational database and associated tools were used for the database development, such as PL/SQL and Oracle Developer/2000. The database is designed for a UNIX operating system and utilizes several of it's tools for data processing and plotting. The present database is designed to be user friendly. It already stores and retrieves data for several wells in El Salvador geothermal fields. The database can be expanded to store other information not considered in the design. Also, it is flexible and easy to modify, with the possibility of handling substantial amounts of data.

1. INTRODUCTION

The following work deals with the development of a database for geothermal fields in El Salvador. At present almost 50 deep wells have been drilled into the Ahuachapán and the Berlín geothermal fields. Geothermal power plants are already in operation in both areas and substantial expansion in production is under way (Montalvo, 1996; Monterrosa, 1993; Quijano, 1994). The two fields are operated by CEL (Comisión Ejecutiva Hidroeléctrica del Río Lempa), the national power company of El Salvador. As a part of the operation of the Ahuachapán and the Berlín fields, comprehensive downhole and production monitoring takes place. The collected field data are initially stored on sheets of paper, but later on they are computerized, either in standard Windows based spreadsheets or as individual files. Unfortunately, the computer storage so far has been limited to PC computers, often without interconnection. This leads to frequent copying of the collected data from one computer to another and may, in some instances cause diffusion and considerable time consumption in the geoscientific work carried out at CEL. As this case of diffuse databases is not only confined to CEL, an international effort has been carried out in order to define safe and sound database environments in interconnected computer networks. This has led to the definition of relational databases and SQL database language (Van Der Lans, 1988).

Efforts at Orkustofnun - National Energy Authority of Iceland in developing and maintaining a comprehensive geothermal database management system have undeniably helped the author in his work to set up a preliminary database for the El Salvador geothermal fields. Computers are ideal to store such data, and a simple system based on individual computer files or spreadsheets, can readily provide effective and flexible data storage. The data can be manipulated easily, although considerable effort may be needed in order to change the data format, to suit various interpretation programs. In geothermal applications, data analysis has begun to demand more flexibility to compare data from different wells or even in different geothermal fields, which is not easy when data are stored in individual files. A data management system can overcome these problems and provide the required features. There are many methods of data management in use today, ranging from a simple paper-based setup, to a computer network database system. The database management system used here is a computer based relational database.

In the following report a general introduction is given on modern computer networks and how a multi-user network is operated. Some general items regarding databases and their structures are discussed. Finally, a preliminary database for El Salvador geothermal fields is shown as well as some associated tools. Special graph routines are presented and several examples of downhole and time-related plots are given. Following the conclusions are some suggestions regarding future development of this preliminary version of an El Salvador database.

2. COMPUTER NETWORK SYSTEM

Figure 1 shows schematically a typical, up to date computer network. The network units can be divided into two groups, central and peripherical devices. The central units are in principle the heart of the system. They interconnect all the network users and, furthermore, operate and store the geothermal database which must be accessible to every geoscientist working on the system. In the following chapter a brief description is given on the respective units of the computer network on Figure 1.

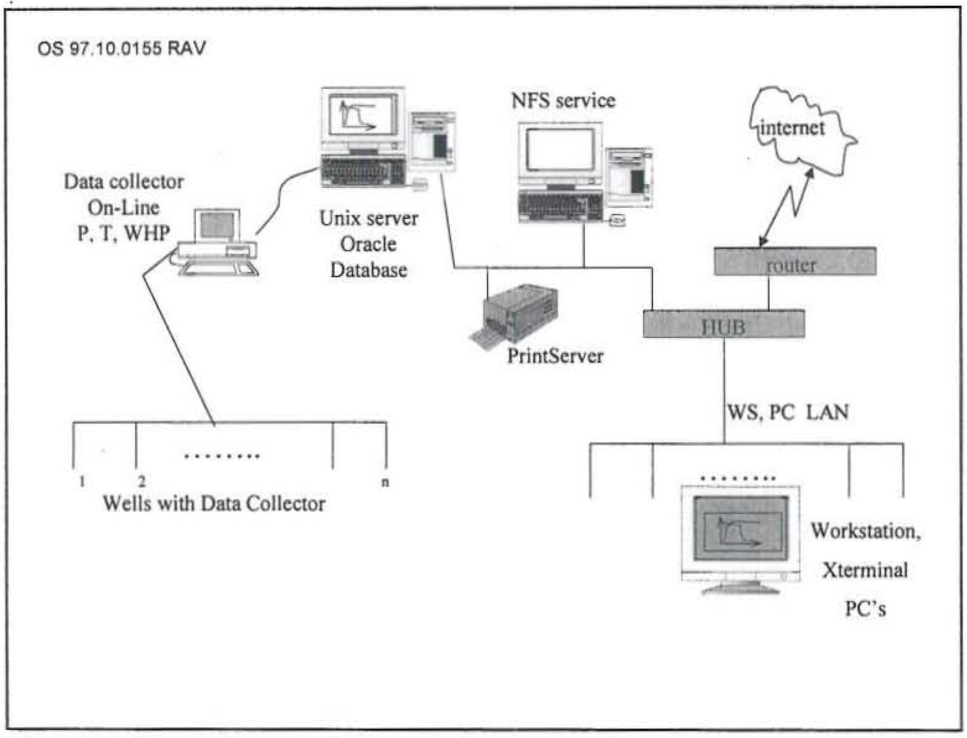


FIGURE 1: A sketch of a modern computer network system

2.1 Central devices

The **server** is, normally, a computer with powerful processing characteristics, with RISC processor, which provides high performance and supports high-bandwidth networking, and enables multiple simultaneous data transfers. The data transfer between processor, memory, I/O and graphics in these servers ais fast. This means vastly improved performance in applications such as database operation and networking.

The **workstations** are similar to the server in capacity, but the difference is often graphic processing in 2D and 3D, using a graphic card. The workstation enables a whole class of applications on the desktop, with X Windows environment, so the user can consult for example the data, draw a temperature profile, and open several windows from different servers.

2.2 Peripherical devices

X terminals are compact desktop units that communicate with, and display the output of multiple networked host computers via workstation-like, high-resolution, multiwindow user interfaces. An X terminal does no application processing but is dedicated specifically to the display and communication of data from the various hosts on the network.

The X terminal is a newcomer on the market of UNIX hardware. An X-terminal is "dumb" as an ASCII terminal, in that without a host computer to connect to, it is nothing but a blank screen with a setup menu. But when properly configured, the X terminal gives the user all the functionality of a workstation without all of its cost and administrative worries. X terminals are sold with a variety of screen sizes, screen depth, resolution, memory configurations and software.

The **monitor** is the single most important part of the X terminal. The size, resolution, and depth of the monitor have a bigger impact on the perceived quality of the terminal than anything else. Common **screen sizes** for X terminals range from 14 inches to 21 inches diagonally. Some X terminals support a "virtual screen", whereby the screen image is actually larger than the screen itself. The portion of the screen that is obscured can be exposed by moving the mouse onto that area.

The **resolution** of a screen is usually given in pixels. X terminal displays range from 640x480 pixels to 1920x1200 pixels. A pixel is the smallest element of the display that can be addressed. The number of pixels effectively determines how much information can be shown on the screen.

2.2.1 Dumb terminal

When the computer first came into commercial use, it operated in a centralized environment. This was the large "mainframe", which weighed more than a car, and sometimes occupied a substantial portion of a room. The mainframe processed programs fed to it by punch cards, magnetic media and eventually an interactive device called the "dumb terminal". The dumb terminal was developed to allow instant access to programs and information held in the centralized computer. The dumb terminal did nothing but display information from the mainframe and accepted information from the keyboard. It had no CPU or memory, and hence did not process information at all.

2.2.2 PCs based clients

The personal computer (PC) or IBM compatible, is a machine with CISC processing, on the market called Intel or Pentium. These machines have spawn in the last year, and have high performance, support high quantity of memory, hard-disk and a wide number of peripherical devices that can be connected to the computer, such as CD-ROM readers, fax/modems, digitizing tables, scanners and data-loggers. But in order to connect the computers to a network it is necessary to install a special software and network interface card, which allows communication between the servers and other computers. These computers work with software such as MS-DOS or Microsoft Windows environment in local mode.

2.2.3 TCP/IP protocol

The TCP/IP is one of a group of protocols developed to interconnect nets of computers. These protocols have a stratified architecture. The first level is the protocol needed to send a packet which identifies the sender, the receiver and the context of the packet. Above this level there is TCP/IP, the protocol that sees that data forwarded will effectively reach the target. For this purpose the message is split up into datagrams and TCP/IP ensures that the datagrams correctly reach the target. Above these levels is still another protocol, IP, which cares for data routing on the Internet. Lastly, there is a fourth level, the interface, e.g. Ethernet, serial port, etc.

2.2.4 Printers and print-server

One of the easiest-to-use and most practical services a network provides is access to shared printers. A printer is probably the first shared peripheral the user installs when a network is set up. Shared printing has some compelling benefits, such as:

- Instead of providing each user with a personal printer, it is better to provide a few high-quality printers and consolidate the expenditure. A network can make it economically feasible to install a high-speed laser printer because the cost of that printer can be spread over many users.
- Sharing makes it possible for network users to have access to a wider variety of printers, for instance, if the user needs to print a graph in full colour on a printer that is connected to the network.
- With sharing printers it is easy to connect a printer to the parallel or serial port of a server, giving all network users access to the printer. This also enables the user to share printers connected to workstations.

2.2.5 Hub and routers

Today local area networking is a shared access technology. This means that all of the devices attached to the Local Area Network (LAN) share a single communications medium, usually a coaxial, twisted pair, or fiber optic cable. Several computers are connected to a single cable that serves as the communication medium for all of them. The physical connection to the network is made by putting a network interface card inside the computer and connecting it to the network cable. Once the physical connection is in place, it is up to the network software to manage communication between stations on the network.

In a shared media network, when one station wishes to send a message to another station it uses the software in the workstation to put the message in an packet. This packet consists of message data surrounded by a header and a trailer that carry special information used by the network software to send data to the destination station. One piece of information placed in the packet header is the address of the destination station.

An important part of designing and installing a LAN is selecting the appropriate medium and topology for the environment. Ethernet networks can be configured in either a bus or star topology, and installed using any of three different communication media.

Coaxial cable was the original LAN medium and it is used in what is called a bus topology. In this configuration, the coaxial cable forms a single bus to which all stations are attached. This topology is rarely used in new LAN installations today because it is relatively difficult to accommodate adding new users, or moving existing users, from one location to another. It is also difficult to troubleshoot problems on a bus LAN unless it is very small.

The star topology LAN is a more robust topology. In a star topology, each station is connected to a central wiring concentrator, or hub, by an individual length of a twisted pair cable. The cable is connected to the station's network interface card at one end and to a port on the hub at the other. This allows all stations to see each packet sent on the network, but only the station a packet is addressed to pays attention to it.

An individual LAN is subject to limits on such things as how far it can extend, how many stations can be connected to it, how fast data can be transmitted between stations, and how much traffic it can support. If a company wants to go beyond those limits, to link more stations than that LAN can support, it must install another LAN and connect the two together in an internetwork.

There are two main reasons for implementing multiple LANs and internetworking them. One is to extend the geographic coverage of the network beyond what a single LAN can support, to multiple floors in a building, to nearby buildings, and to remote sites. The other key reason for creating internetworks is to share traffic loads between more than one LAN. A single LAN can only support so much traffic. If the load increases beyond its carrying capacity, users will suffer reduced throughput and much of the productivity achieved by installing the LAN in the first place will be lost. One way to handle heavy network traffic is to divide it between multiple internetworked LANs.

The term internetworking refers to linking individual LANs together to form a single internetwork. This internetwork is sometimes called an enterprise network, because it interconnects all of the computer networks throughout the entire enterprise. Geographically distant company sites can also be tied together in the enterprise-wide internetwork.

There are three major types of devices used for internetworking: routers, bridges and switches. Currently, the most used internetworking devices are high-speed routers, especially in wide area internetworks, linking geographically remote sites. Routers link different LANs or LAN segments together. Routers were originally designed to allow users to connect these remote LANs across a wide area network and to determine the best route through a complex internetwork. By placing routers on LANs at two distant sites and connecting them with a telecommunications link, a user on one of the LANs can access resources on the other LAN as if those resources were local. Because routers are complex internetworking devices, they use different Network Layer Protocol Information within each packet to route it from one LAN to another.

2.3 Communication between clients and servers

2.3.1 UNIX operating system

An operating system is a set of programs which permits a computer to manage the flow of information between its processor, memory, peripherals, screen, hard disk and printer. UNIX is a text-based operating system that has capabilities of a graphical user interface. It controls the resources of the computer (processor, memory, disk and devices) and provides services to programs which run under it. These programs in turn provide services to users. Unix is a multi-user and multi-tasking operating system with built-in security. Multi-user means that there are different users working at the same time. Multi-tasking means that one computer can perform several unrelated tasks at the same time. As a corollary to this, the Unix security features allow users to protect their resources or to allow access to other users.

Unix is also a cross-architecture operating system. This means that it works the same way across many different types of computers, with different innards. Unix is presently the only operating system which runs on many different hardware architectures.

2.3.2 X-Windows

The X Windows system, called X or X11 for short, is a network-based graphics Windows system (Mui and Pearce, 1993). X is based on the *client-server model*, in which the application program (the *client*) does not directly access the display, but communicates with an intermediary display program (the *server*).

One important feature of the client/server model is that the client and the server programs can communicate over the network. They do not need to run on the same machine, or even in the same building. This mean that an X display is an ideal front end for a distributed computing environment. A system administrator might open windows on each of several machines that he/she is maintaining. The client and the server communicate using the X protocol, which can run on top of a UNIX domain, TCP/IP, or DECnet. Technically, the X protocol is the true definition of X. However, when referring to X, it not only means the protocol, but also the widely available implementation of clients, server and libraries that use that protocol.

Since the client and the server can run on different machines, the local display machine can get away with running a server program and nothing else. X servers can run on single-tasking DOS-based PC's, which connect across the network to multi-user systems capable of running multiple graphical applications. More importantly, this feature has led to the development of low-cost *X terminals*, designed specifically for running X servers. Using X terminals, a company can give multiple users the ability to run graphic-intensive programs, without having to buy for each user a machine powerful enough to execute the graphic programs themselves.

X has a great potential because it can be ported to any architecture, operating system or display type. Servers have been written for all sorts of architectures, under all sorts of operating systems, for all types of displays. The only requirements are that there is a keyboard, a graphics display and an input pointer (such as a mouse). And because the server handles the hardware and operating-system dependencies, client programs can be almost completely portable.

Currently, most client programs run on some flavour of the UNIX operating system, but they have also for a long time been available under many other operating systems (such as VMS, Lynux, SCO UNIX), and recent products now run X clients (as well as servers) on Windows based PC's. Furthermore, clients have been written to be heavily dependent on programming libraries. This mean that once the libraries are ported to another operating system, clients using those libraries should be easily ported as well.

Client-Server terminology often seems "backward" to people who are new to X. Since the X display runs on a local machine on the user's desk, one might think that the X display should run the X *client* program. The X server is a *display server*. It makes the keyboard and display available to applications running on the other machines across the network. The fact that the user can, and often does, run clients locally, does not make the display server any less a server. Clients must still connect to it to make use of the services (display and keyboard) it manages.

The X display server accepts connections from any number of X application clients. These clients might run on any machine on the network. An X server can be written for any graphic display. These displays, each consisting of a pointing device, a keyboard and at least one monitor, can differ in several ways.

- Monitors have different screen sizes and different resolutions. Some monitors have colour support. A server might support anything from 1 to 24 bits of colour per pixel.
- The pointing device might be a mouse. Most displays use a mouse as a pointing device, but the mouse might have 1, 2, or 3 buttons.
- Different keyboards have different layouts, and each key generates different control sequences. The user can depend on alphanumeric keys on U.S. or European keyboards, but the user cannot depend on there being function keys, an ALT key, or numeric keypad.

On other windows systems, one might be able to configure this information directly into the application at installation time, since there is only one display that the program can access. X clients, however, must be able to run with all possible servers. The X server, therefore, needs to mediate between clients and the specifics of the display.

For the output device, i.e. the monitor, the server not only needs to know how to draw to display as specified by the client, it also needs to be able to tell the client the screen dimensions or whether colour is supported. If a user has more than one monitor, each monitor can be used as a separate screen of the display. Input devices, the mouse and keyboard, can also differ. In order to insulate clients from these differences, the server maintains a mapping between physical buttons and keys and corresponding logical identifiers.

2.3.3 Windows PCs based

Windows PCs based, normally Windows 95, will present a major step forward in functionality of desktop and portable PC platforms, by providing a system that is even easier, faster and more powerful to use, and which maintains compatibility with the Windows-95 and MS-DOS operating system-based applications and hardware peripherals in which customers have invested.

2.3.4 NFS software

The Network File System (NFS) service enables computers of different architecture, running different operating systems, to share file systems across a network (Hewlett Packard, 1994). The NFS service makes the physical location of the file system irrelevant to the user. The user can use the NFS service to enable users to see all the relevant files regardless of location. Instead of placing copies of commonly used files on every system, the NFS service enables you to place one copy on one computer disk and have all other systems access it across the network. Under NFS service, remote file systems are almost indistinguishable from local ones.

The Network File System allows a client node to perform transparent file access over the network. By using NFS, a client machine operates on files residing on a variety of servers and server architectures, and across a variety of operating systems. The file system on the remote machine does not have to be compatible with the local file system.

The terms client and server are used to describe the roles that a computer plays when sharing file systems. If a file system resides on a computer's disk and that computer makes the file system available to other computers on the network, then that computer acts as a *server*. The computers that are accessing that file system are said to be *clients*. The NFS service enables any given computer to access any other computer's file systems and, at the same time, to provide access to its own file systems. A computer can play the role of client, server, or both, at any given time on a network. A server can provide files to a diskless client. A diskless client relies completely on the server for all its file storage, and it can only be a client, never a server.

Clients access files on the server by mounting the server's shared file systems. When a client mounts a remote file system, it does not make a copy of the file system; rather, the mounting process enables the client to access the file system transparently on the server's disk. The mount looks like a local mount and users type commands as if the file systems were local.

The objects that can be shared with the NFS service include any whole, or partial directory trees, or a file hierarchy-including a single file. A computer cannot share a file hierarchy that overlaps one that is already shared.

In most UNIX system environments, a file hierarchy that can be shared corresponds to a file system or to a portion of a file system; however, NFS support works across operating systems, and the concept of a file system might be meaningless in other, non-UNIX environments. Therefore, the term file system is normally used for a file, or file hierarchy, that can be shared and mounted over the NFS environment.

2.3.5 PC-X server

For some networks, the local processing capability of a PC is important. In those cases, PC X software provides the PC with X terminal functionality and the choice of Windows or Motif for its 'look and feel' environment. PC X software provides open access and freedom of choice in every area: host computers, applications, network protocols and desktop devices. Because X functions independently it can operate with any computer regardless of its operating system.

3. DATABASE INTERNAL STRUCTURE AND DEVELOPMENT

The concept of databases has evolved rapidly during the last decade and is now days a well defined object in modern computer systems. In the following chapter a brief introduction is given on the various types of databases (Singh et al., 1997). The text is however inclined towards solutions of the Oracle database system, as the author was trained in its use during his stay in Iceland.

3.1 Hierarchical databases

A hierarchical database model stores data in a tree-like structure, with parent and child relationships between the records in the database (Figure 2). The hierarchical method has a number of drawbacks. Before the information of a child record can be accessed, the parent record must be accessed. To overcome this limitation, much repeated and redundant data is normally stored in the database. In addition, a child record can only relate to one parent. Consequently, if the user wants two model relationships to a child, it is necessary to introduce additional data to show the extra relationships. Another major drawback of the hierarchical model is that relationships between data are hard-coded into the database. When the user sets up the database, the user specifies how the parent and child rows relate, using some of the fields as key values.

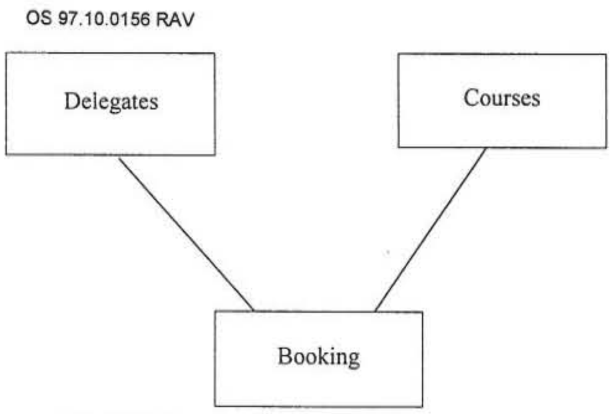


FIGURE 2: An example of hierarchical database structure

3.2 Network databases

The network database model can be seen as an extended version of the hierarchical model; it was introduced to overcome some of the limitations of the hierarchical model. The major difference between the two models is that in the network model, a record can have predefined relationships to many other records, not just to one parent record. The network model includes two sets of objects: the records themselves and the links between the records. The user specifies the way in which one set of records relates to another when the user sets up the data structure, so it is difficult to change that relationship later. This difficulty stems from the fact that the relationship is hard-coded as part of the database structure and cannot be specified on-the-fly. Figure 3 illustrates a network database.

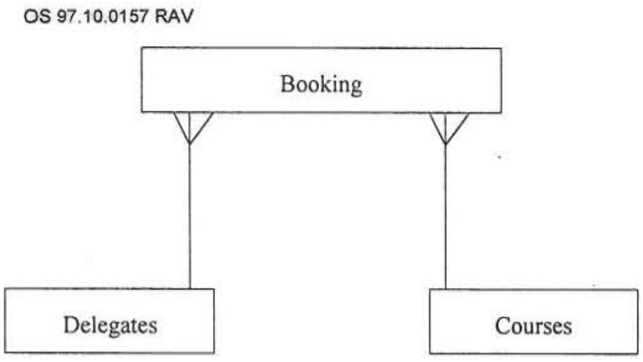


FIGURE 3: An example of network database structure

3.3 Relational database

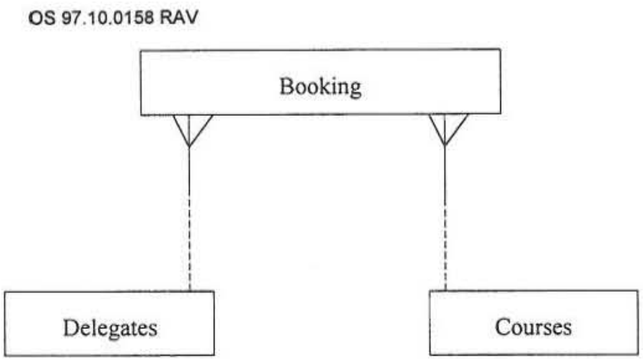


FIGURE 4: An example of relational database structure

The relational database model attempts to overcome some of the failings of the hierarchical and network databases and provides easy-to-use and flexible data structure. Figure 4 shows an example of table structure in a relational database. The relational database model includes data structures as tables, operators as the SQL language that can be used to manage the data structure, and some integrity rules as constraints that ensure that the data obeys the rules defined by the system.

The relational model is based on relational algebra concepts and theory, and the early versions used some of the same terminology to describe the processing that could be performed on the data structures. Another important concept in relational databases is that relationships between tables are not hard-coded in the structure of the data; that is, there are no pointers in the data to relate one table to another. This means that the user can specify the relationships between two or more sets of data at development time rather than when the tables are first created. This greatly improves the flexibility of the database management system.

3.4 Database as a storage device

A database is an integrated collection of stored data and combinations of data can be stored, retrieved, updated, added or deleted, quickly and easily. The usefulness of any database management system depends on its capability to store information efficiently, and its efficiency in accessing and manipulating the data in the database. In this project, the Oracle relational database management system was used to provide the structure and mechanism to process a vast array of information (Abbey and Corey, 1995). The mechanism to define and access data is the Structured Query Language (SQL). The SQL is a language standard used to interface with relational databases, and is used by developers and database administrators to perform commands and queries to manipulate an entire set of rows at a time, rather than by processing one row at a time. Another tool used for managing data was Oracle Loader, but this tool allows moving data from other external data source, such as ASCII files, into an Oracle database. To extract data from Oracle's database the sqlpp tool was used, an object oriented C++ Class Libraries for relational database application from Rogue Wave Software (1994). It allows SQL queries to be entered from the command line, it is easy to use and retrieves data very quickly.

3.5 Software application development environment

Developer/2000 is an Oracle software development environment that runs as a client/server application. The client, which can be Windows PC based, or using a Unix variant running OSF/Motif, talks to a Developer/2000 server, which then talks to the database server. Each client in its platform retains the Windows look and feel of the native operating system environment.

It achieves this ease of use by being strictly a client/server application. The user manages the user interface, and the Developer/2000 server handles the remaining functions, including verification, validation and system rules that were defined in the relationships of the tables, with Developer/2000 as developer tier. The server tier may be a database product, but it doesn't have to be. This software can talk to all major databases.

Developer/2000 was used to create graphical user interface for input, retrieval and maintenance of data, on the screen. It allows form creation, providing operators access to information stored in a database. A form is more than a single window, it can include database table items displayed in multiple windows, making it very friendly for the end-user.

3.6 Graphic tools to manipulate data

To plot the data retrieved from Oracle's database the Geographical Map Tools (GMT) was used (Wessel and Smith, 1991 and 1995a). GMT is a collection of public-domain Unix tools that allows one to manipulate (x,y) or (x,y,z) data sets (filtering, trend fitting, gridding, projecting etc.). It produces PostScript illustrations ranging from simple x-y plots, via contour maps, to artificially illuminated surfaces and 3-D perspective views in black/white or 24-bit colour. The processing and displaying routines within GMT are completely general and will handle any (x,y) or (x,y,z) data as input.

Another tool used was 'tp', which is a program by Thordur Arason at Orkustofnun. It plots dates versus value. This tool is very flexible and can treat input data in several date formats and produces either

Postscript or Tektronix output files. Also heavily used in the shellscripts is the AWK Programming Language of UNIX (Aho et al., 1988).

To display the graphs one can use GhostScript and GhostView. GhostView creates the viewing windows and allows selected pages to be viewed or printed, and GhostScript is an interpreter for the PostScript description language that draws the file in GhostView. The source code for these tools is available free on the internet (Wessel and Smith, 1995b).

4. DESIGNING DATABASE FOR GEOTHERMAL SYSTEMS

4.1 The golden rules of the study

The design and implementation of a database system must take into account the normalization rules. The diverse data typically collected in a geothermal field, such as, downhole temperature and pressure, or wellhead steam and water flow, demand the use of a data management system based on relational databases. The system should facilitate the storage, retrieval, interpretation and comparison of data, and at the same time minimize errors and misdiagnosis, especially typographical errors. Ideally, the system should include the following list of properties (Anderson, 1995a and 1995b; Anyal, 1995; Bertani, 1995; Salvania, 1995):

- *Data collection.* The data collected should be of good quality and reliability. The quantity of data may need to be reduced by filtering to avoid overloading data storage capability.
- *Data storage.* An appropriately structured database should be available for each data set to be entered. All data entered in the system should include the measurement location, the measurement conditions, measurement units and measurement errors if available.
- *Data retrieval and calculations.* All data should be accessible to all users involved in the geothermal program. Appropriate methods should be available to access data stored in other locations. Selection of data for maintenance purposes should be more tightly controlled to avoid accidental modification or deletion.
- *Data maintenance.* The data management system will maintain the links between each data record and the related data. For instance, all well measurements will remain linked to the appropriate well, even if the name or the location of the well is changed. Appropriate backup mechanisms should exist to prevent accidental loss of data caused by computer problems.
- *Data manipulation and visualisation.* The data management system should include the appropriate programs to analyse and interpret data sets. The system should correctly format the data selected for input to the programs. A wide range of graphical methods should be available to display the data. These include X-Y plots of any two or more measurement values. All graphs should display information on the data source, the measurement units and any scaling that may be applied, and suitable comments. It should be possible to print both data and graphs, in order to produce a permanent record.

4.2 The database tables

In the design of a database, it is necessary to consider the integrity and uniqueness of the data. The relational model allows creating relationship between the various data; this is achieved by creating orderly organised tables. When there is a main object in the database, such as a well, it is necessary to associate its information, for instance a unique well_id number, well name, X and Y coordinates, elevation, maximum depth, measurement quality in the X-Y coordinates and in elevation and an area name identification.

For this purpose a number of tables were created to store selected geothermal data in El Salvador. Figure 5 shows a schematic layout of the table structure and their connections. The tables are *well, xy_quality, zquality, casing, log, series, probe, logtype, tool, logdata* and *seriesdata*. The table names are chosen in order to explain the content of each table. The column names with its properties were defined, and relationships among the tables were set. Primary keys were also selected. For easy identification of the keys in Figure 5, they are separated with a line between the primary key and the other columns. Primary keys may be comprised of several columns. This is to prevent any record from storing similar and inaccurate information. As an example a measure in the *logdata* table has the log_id and the depth as the primary key. The primary keys are also used to connect information from one table to another.

A brief description of the data tables in Figure 5 is given below. Appendix 1 gives a full listing of the queries that were used to create the table structure.

Well: This is the central table of the database and stores the most important property, namely the well_id numbers. This is a unique number and refers to only one well. Other properties are the well name (AH-14), its X, Y and Z locations, a quality measure on the well coordinates, the maximum depth of the well and the area_id number.

XY_quality and **Z_quality**: These tables are near identical except that the xy_quality table stores information on the accuracy on the two surface coordinates, whereas the z_quality table provides the accuracy in elevation. In both cases the quality key is simply a unique number and refers to 1) an error range in the respective coordinate and 2) a description on the quality. It should be stressed that the database does not provide any information on the coordinate projection or distance and elevation units. In future work, attention should be paid to this item and preferably, standardized such that the coordinates refer either to a national coordinate grid of El Salvador or simply to the standard geographical units (latitude, longitude).

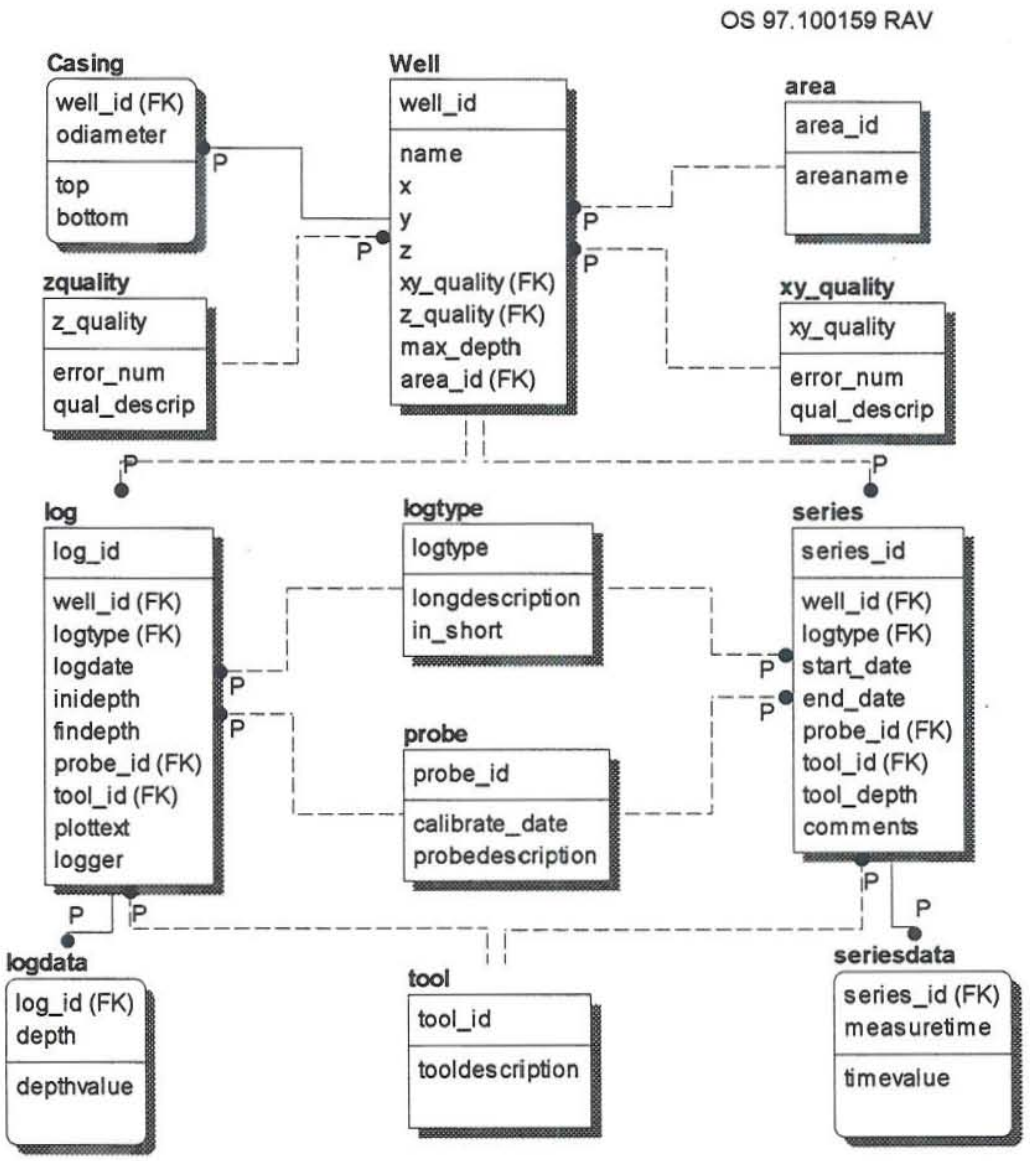
Casing: Provides information on well casings. The primary key is the well_id and the odiameter. Given the well_id one can either insert well casings in new wells or retrieve casing information on older wells (diameter, top and bottom depth).

Log: This table is intended to give information on all logging jobs in a well with given well_id number. The two foreign keys well_id and logtype define the well and the type of measurement (pressure, caliper, etc.). Also given are the date of measurement, the depth range, the logger initials and a special plot text which may be called upon when making graphs with the downhole data. When all this information has been provided, the database will commit them and store them as a new record in the *log* table with a unique log_id number.

Logdata: When a log_id number is available, one can store the associated data string of depth, depth value into the table *logdata*. The bulk of the downhole information is therefore stored in this table.

Series: This table is similar to the *log* table, except that this time the *series* table is provided. As before are seen the foreign keys well_id, logtype and tool_id. Also given are the start and end dates of the serie, the tool depth (equals 0 if wellhead property) and some comments on the measurement. When all this information is committed, a unique series_id number is created by the Oracle system.

Seriesdata: Is similar to the *logdata* table. It is stores the series_id, the measure time and the respective time value (temperature, pressure, etc.).



FK: Foreign key meaning that the key is defined in another table

P: One to one to many relationships

——————— Relationship propagates from parent primary key to primary key of the child

— — — — Relationship propagates from parent primary key to non-primary key of the child

FIGURE 5: A schematic layout of the database table structures and their connections

Logtype: Defines the type of log under consideration (temperature, caliper, etc.). This is a character. The logtype is either described in long or in short, depending on its use. (The table provides description of type of log in short format for use in tabular presentations, as well as in long format.)

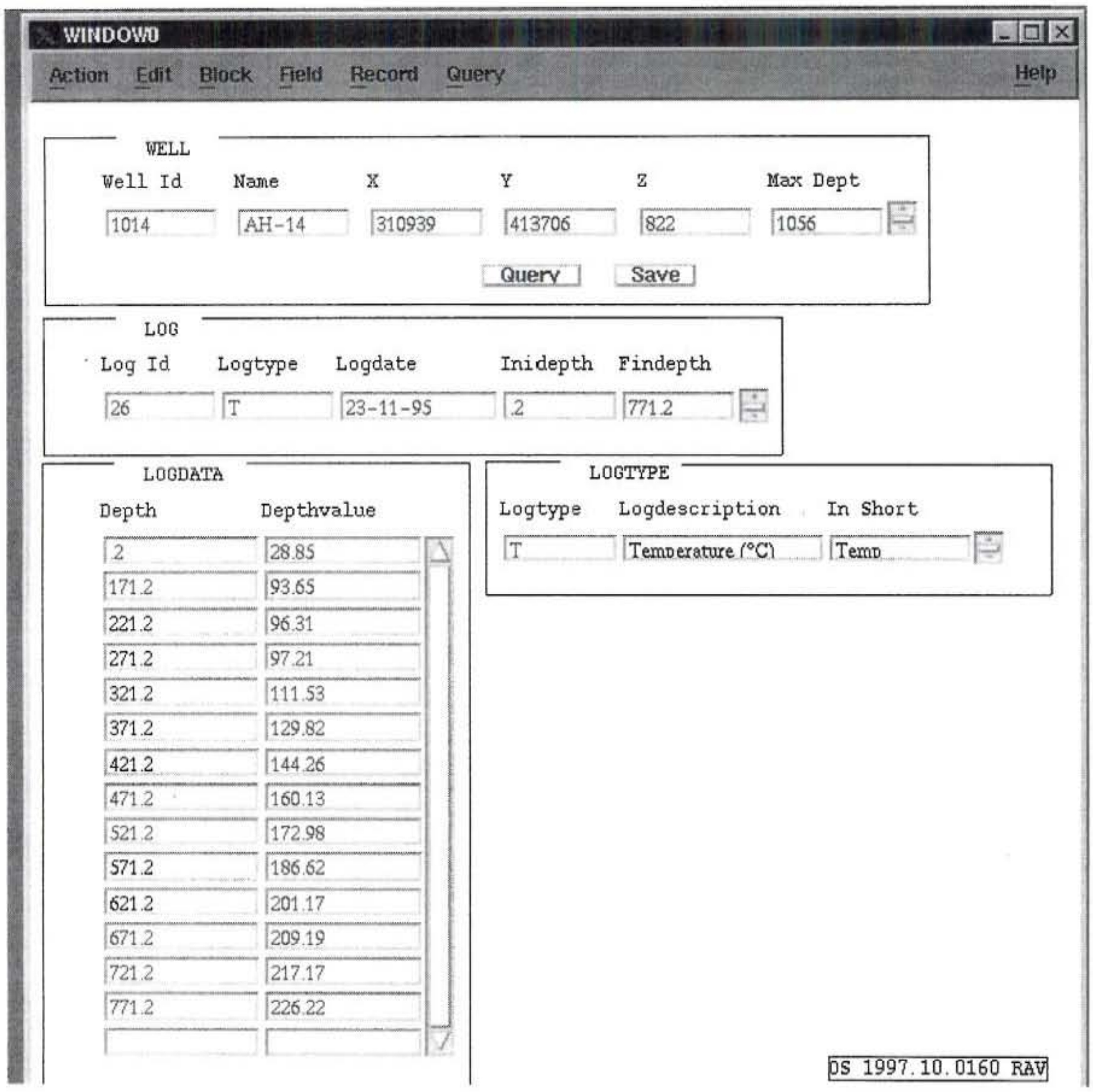
Tool: Holds information on the various tools that are in use in the geothermal logging such as a logging truck_id. The table columns are simply a unique tool_id number and its description.

Probe: Is similar to the *tool* table, except that this time the table stores information on the measuring probes in use. As an example a number of mechanical probes have been used to log downhole pressure in Ahuachapán, although the same tool (logging truck) was used to run the logs. Strictly speaking the probe definition refers to the individual probes that are attached to the cables of logging trucks, but is here also assumed to cover permanent pressure, temperature or flow sensors that are attached to a wellhead.

The table design presented in Figure 5 is not complete. One may point out things like a missing comment column in the *log* table, or a column showing the measure unit in the *logtype* table. These are easy to improve simply by adding columns to the respective tables, using standard database commands. The standard precaution is that the new column may only have reference to the table under consideration, in order to keep the database relational and normalized. This kind of work should be carried out in El Salvador with the cooperation of the people who will benefit from the existence of the database.

4.3 Forms and queries

Figure 5 presents the table structure that was adapted to store some of the geothermal data which has been collected in El Salvador. In general this is a work of the system administrator, conducted however, in close cooperation with the people who will use the database.



WELL

Well Id	Name	X	Y	Z	Max Dept
1014	AH-14	310939	413706	822	1056

LOG

Log Id	Logtype	Logdate	Inidepth	Findepth
26	T	23-11-95	2	771.2

LOGDATA

Depth	Depthvalue
2	28.85
171.2	93.65
221.2	96.31
271.2	97.21
321.2	111.53
371.2	129.82
421.2	144.26
471.2	160.13
521.2	172.98
571.2	186.62
621.2	201.17
671.2	209.19
721.2	217.17
771.2	226.22

LOGTYPE

Logtype	Logdescription	In Short
T	Temperature (°C)	Temp

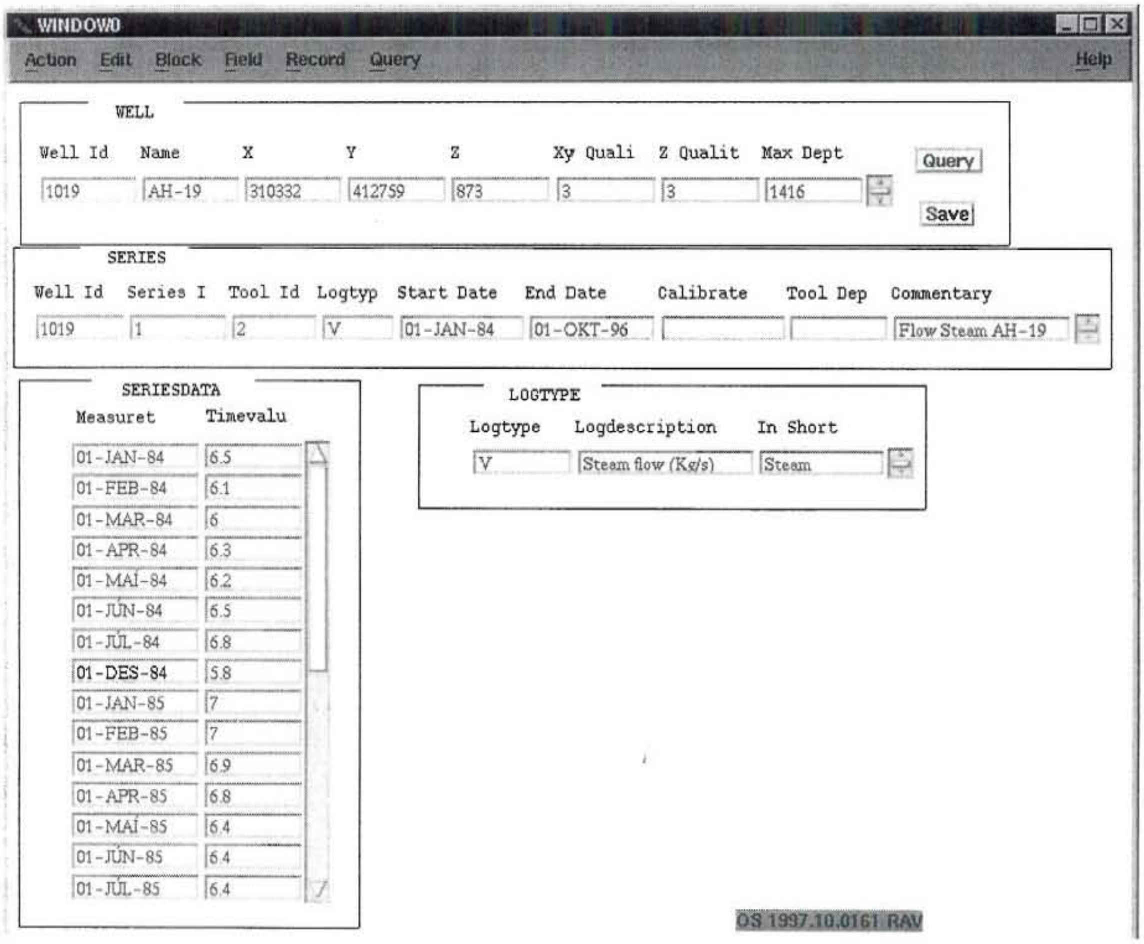
FIGURE 6: An Oracle form for working with downhole data

The next step is then to set up an environment to insert and retrieve the stored data. For this purpose two important tools are commonly used in the Oracle system, *forms* and *queries*. The forms are used to insert new records and to take a quick look at the stored data. The queries are used to retrieve the data, either within the forms or by using SQL commands.

The forms used here were created using Developer/2000. On the terminal they appear as a group of blocks, that contain items from a number of tables. Each block is an interface between the form and the database. In these forms the user can access the databases for several purposes, such as retrieve records from the database, change data of those records, add new records or delete unwanted records. Figure 6, shows as an example a form used to access the data tables *well, log, logdata* and *logtype*. Each of the four blocks provides access to the respective table. By placing the cursor within one of the blocks, one can find data by working on the window menu and the scrollbar. This form is intended for working on downhole data in geothermal wells. Figure 7 shows a similar form but this time concentrated on data collected with the time.

5. USING THE GEOTHERMAL DATABASE

The tables and forms presented in Chapter 4 apply for inserting and viewing the stored data on the terminal. In the following some examples are given on how to extract and plot the data. For that purpose two UNIX shell scripts were developed, *goragraf* and *seriegraf*. Both scripts utilize standard shell tools, the GMT graphics and the SQL environment. These scripts are similar to tools that are presently in use at Orkustofnun, but have been modified in order to be operational when back in El Salvador.



WELL

Well Id	Name	X	Y	Z	Xy Quali	Z Qualit	Max Dept
1019	AH-19	310332	412759	873	3	3	1416

SERIES

Well Id	Series I	Tool Id	Logtyp	Start Date	End Date	Calibrate	Tool Dep	Commentary
1019	1	2	V	01-JAN-84	01-OKT-96			Flow Steam AH-19

SERIESDATA

Measuret	Timevalu
01-JAN-84	6.5
01-FEB-84	6.1
01-MAR-84	6
01-APR-84	6.3
01-MAÍ-84	6.2
01-JÚN-84	6.5
01-JÚL-84	6.8
01-DES-84	5.8
01-JAN-85	7
01-FEB-85	7
01-MAR-85	6.9
01-APR-85	6.8
01-MAÍ-85	6.4
01-JÚN-85	6.4
01-JÚL-85	6.4

LOGTYPE

Logtype	Logdescription	In Short
V	Steam flow (Kg/s)	Steam

FIGURE 7: An Oracle form for working with time related data.

5.1 Downhole pressure and temperature data

The script *goragraf* was developed to plot downhole pressure and temperature data from the database. Goragraf is a shell script using korn shell syntax (Bolsky and Korn, 1995). It only requires the well identification number (well_id) to draw all or selected downhole profiles, using SQL commands to select and get the data from the Oracle database, and GMT to produce a Postscript output which is viewed with GhostView. Several options are available to select graphic attributes such as; plot pressure instead of temperature, use elevation instead of depth, plot selected profiles, change graph and text sizes, etc. Appendix 2 gives a list of these options.

The basic way to produce graph is to write on the command line:

```
goragraf -w #
```

where # is the representative well id number. In this work, wells in Ahuachapán hold a well_id number between 1001 and 1032, and in Berlín between 3001 and 3010. In order to draw a graph, it is recommended to check first how many measurements are available by typing:

```
goragraf -w 3002 -L
```

In this case we want to examine temperature data in well TR-2 in Berlín (-w 3002) and only get a list of available measurements on the terminal (`-L`). This gives:

```
      Get the data available in ORACLE database for well_id = 3002...
                Get the data for the well TR-2 from ORACLE...
        No  Log_id  Date(d-m-y)   Type  Depth_min  Depth_max  Comment
-----------------------------------------------------------------------
         1    246    28  1 1982     T          0       1902
         2    247     5 12 1994     T          0       1802
         3    248    14  6 1996     T          0       1802
-----------------------------------------------------------------------
```

The next step is to get a graph of the data by displaying the command:

```
goragraf -w 3002 -e -H 'Berlín/Temperature Profiles/Tr-2'
```

The result is Figure 8, showing 3 downhole profiles, their legend and a header. Note that the flag `-e` gives the depth in terms of elevation (the Z-coordinate is provided in the table well).

Figure 9 shows all collected downhole temperature data in well AH-14 in Ahuachapán. A total number of 33 profiles is shown. The line and symbol type are selected by default in the *goragraf* shell. Due to the high number of profiles, plotting legend for them all is practically impossible. By adding `-u` to the plot command, one can skip the legend, as shown below:

```
goragraf -w 1014 -u -H 'Ahuchapán/Temperature Profiles/Well AH-14'
```

Figure 10 shows a graph of all available pressure data in well AH-21. As before, no legend is shown. The flag `-P` selects pressure data instead of the default temperature data option. The flag `-X` and `-Y` control the graph size. The plot command is then:

```
goragraf -w 1021 -P -u -e -X 10 -Y 10
```

Note that one of the pressure profiles on Figure 10 has a negative gradient indicating that an error was done during the data insertion. This demonstrates that good graphic tools also help in keeping the database correct.

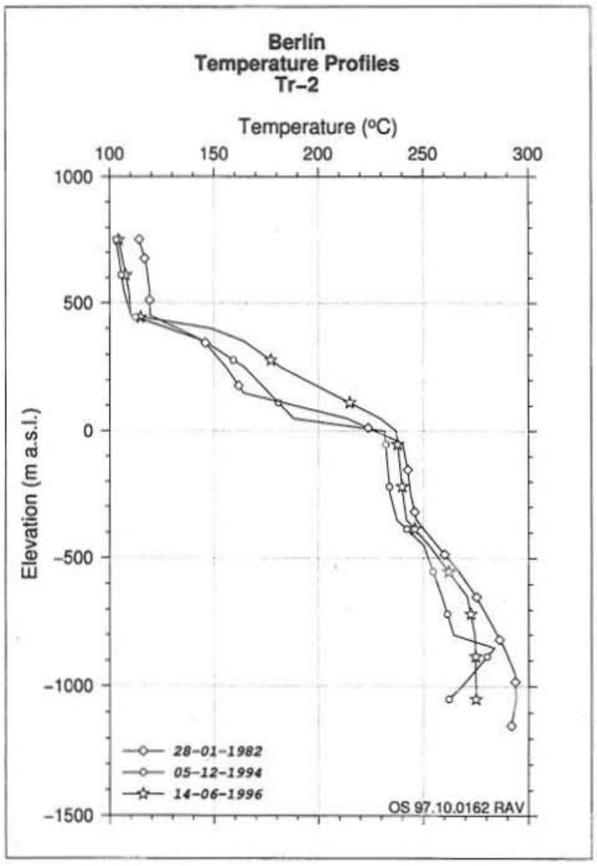


FIGURE 8: An example of downhole temperature data from well Tr-2 in Berlín

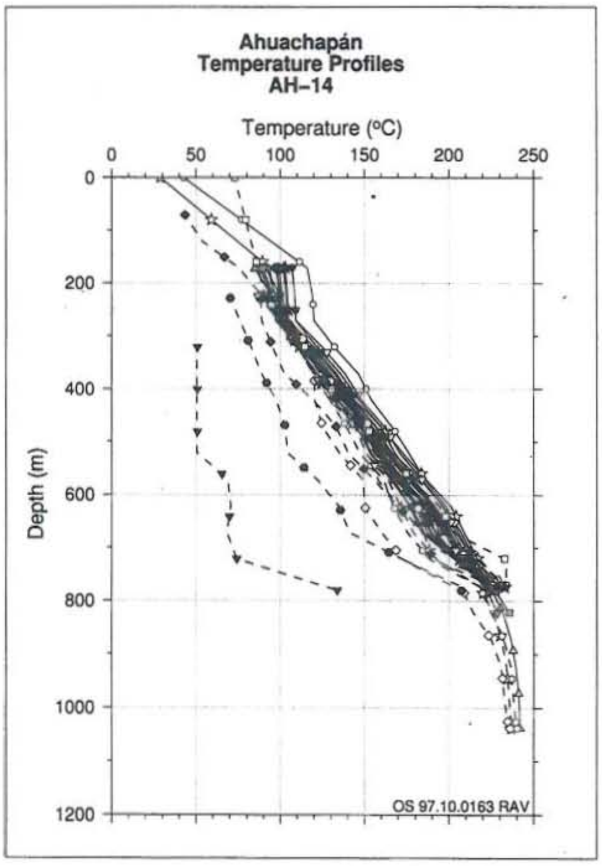


FIGURE 9: An example of downhole temperature data, well AH-14 in Ahuachapán

It should be noted that the graphs on Figures 8-10 show only an example of the capabilities of *goragraf*. Furthermore, it should be mentioned that the tables in Figure 5 hold information on other plot parameters such as a default figure header (areaname in *area* and name in *well*). The casing information in the *casing* table can also be used to plot the well design next to the temperature graph. All of this is achieved by knowing only the well_id number. This should demonstrate how powerful a well defined, relational database can become in an adequate computer environment.

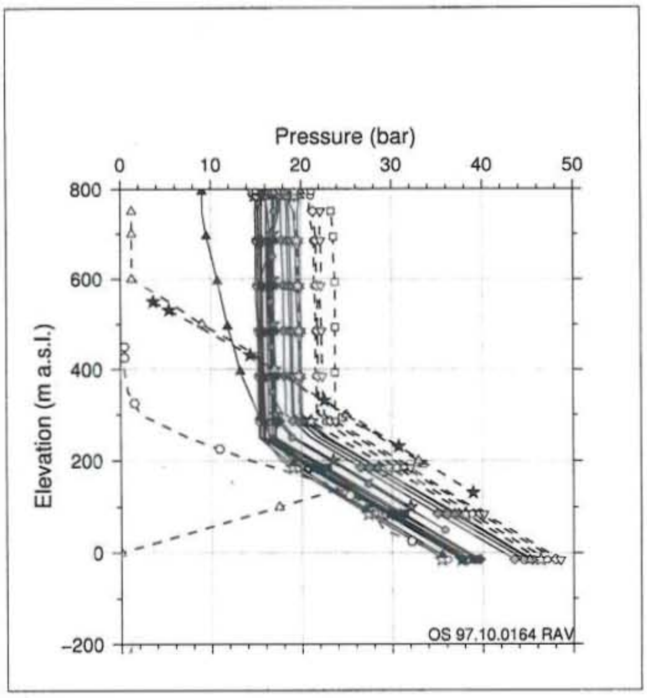


FIGURE 10: An example of downhole pressure data from well AH-21 in Ahuachapán

5.2 Production data

The shell script *seriegraf* works similarly as the goragraf script. This time however, only a preliminary option was designed. The script works as follows:

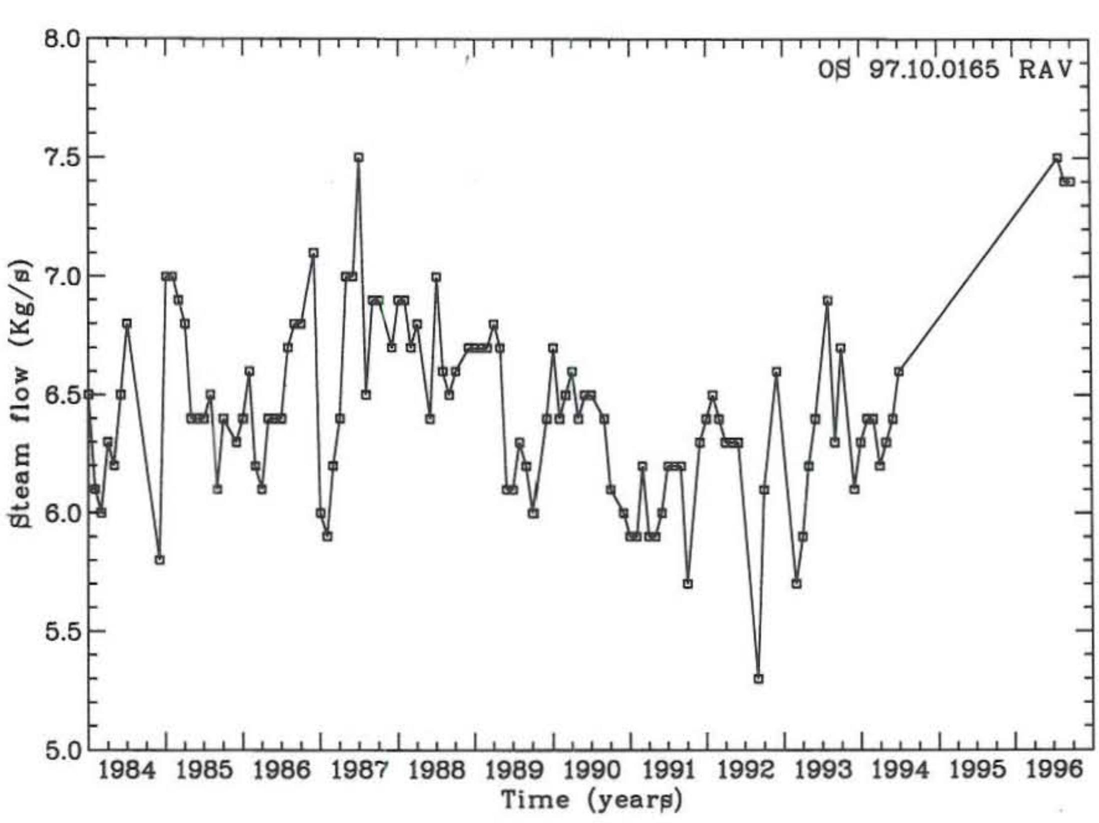


FIGURE 11: An example of steam flow with time from well AH-19

```
seriegraf # '*'
```

where # is the well_id number and the logtype is given by '*', where the character * is provided by table *logtype* (Figure 5). Unfortunately there are no time plots available in the GMT package. The plot program *tp* was, therefore, applied to draw the time dependent data.

Figures 11 and 12 show two examples of using the *seriegraf* shell. Figure 11 presents steam flow from well AH-19 during a 13 year period, plotted by the command:

```
seriegraf 1019 'V'
```

Figure 12 shows finally detailed downhole pressure data of well AH-25, collected by a Sperry Sun data logger. A cycle of 12 hours is evident in the data, reflecting the lunar tides. The Sperry Sun tool appears, thus, to be in healthy condition, at least with respect to small scale pressure fluctuations. The command used to plot the graph is:

```
seriegraf 1025 'P'
```

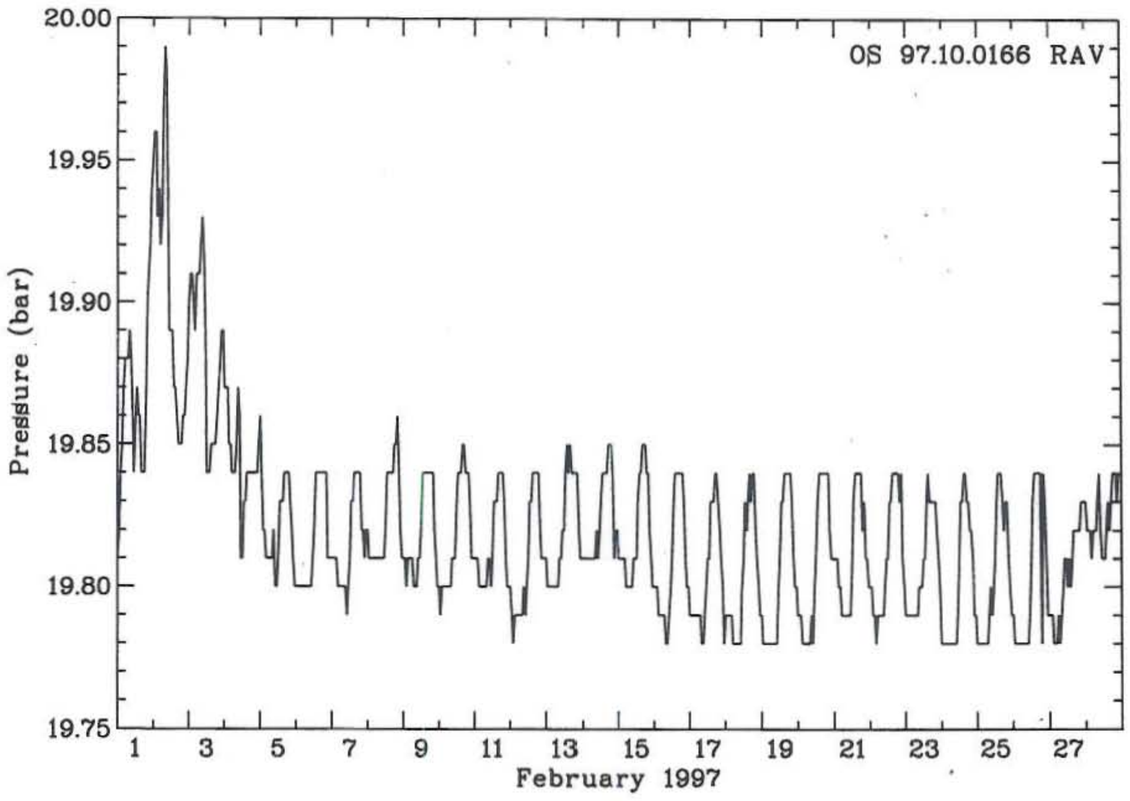


FIGURE 12: An example of downhole pressure with time in well AH-25

6. CONCLUSIONS AND RECOMMENDATIONS

The main conclusions of the report are

- A study of modern computer systems shows that a comprehensive network with powerful servers is essential in order to operate a multi-user database.
- A preliminary database design has been developed for geothermal fields in El Salvador. This database contains thermodynamic information such as downhole temperature and pressure and production monitoring with time.
- This relational database uses the Oracle software for processing and storage. It is flexible and easy to modify, with the possibility of handling great amounts of data.
- Two UNIX shell scripts have been developed to plot the downhole data and for monitoring data. They only require simple flags for the execution.
- The database and the associated tools appear to work adequately and are already easy to use.

The database presented here is only a first step in setting up and operating a full scale database for CEL in El Salvador. A lot of information worth storing has been negleted in the study, geothermal data like fluid chemistry and isotope content, the drill cutting and X-rays analysis, lithological logging in wells and geophysical data. All of these can be added to the preliminary database given that the time, computer environment and cooperation of the people involved are available. The last part may be of most importance since a successful database operation requires substantial group work.

ACKNOWLEDGMENTS

I would like to express my gratitude to the Government of Iceland and the United Nations Geothermal Training Programme for the opportunity to broaden my knowledge in the field of geothermal energy, to Dr. Ingvar B. Fridleifsson, Director of the Geothermal Training Programme for selecting me to attend the 1997 six months training course and for all his help during the course; to Mr. Lúdvík S. Georgsson and Gudrún Bjarnadóttir for their special help and guidance throughout the training period. My sincerest thanks go to my advisors, Mr. Grímur Björnsson, Hilmar Sigvaldason, and Thór Sigurdsson for sharing theirs experience and knowledge, and Sigvaldi Thordarson, for great help and critical advice to expand this report. Also, thanks to all the staff at Orkustofnun for their willingness to help.

I would like also to thank the Comisión Ejecutiva del Río Lempa (CEL), for giving me the opportunity to attend this course.

My deep gratefulness to my family, especially my wife Jenny Lissette and my daughter Mónica Gisselle for giving me their spiritual support during the six long months of physical separation, and my parents Elena Hurtado and Jaime Cabrero.

REFERENCES

Abbey, M., and Corey, M., 1995: *ORACLE: A beginner's guide.* Osborne McGraw-Hill, USA, 522 pp.

Aho, A., Kernighan, B., and Weinberger, P., 1988: *The AWK programming language.* Addison-Wensley Publishing Company, USA, 210 pp.

Anderson, E.B., 1995a: Applying data management to a geothermal project. In: Anderson, E.B. (convener), *Course on data management and related software in geothermal applications.* World Geothermal Congress 1995, IGA pre-congress course, Pisa, Italy, May 1995, 107-115.

Anderson, E.B., 1995b: Geothermal data management case studies: Resource assessment. In: Anderson, E.B. (convener), *Course on data management and related software in geothermal applications.* World Geothermal Congress 1995, IGA pre-congress course, Pisa, Italy, May 1995, 13-28.

Anyal, S., 1995: The technology now. In: Anderson, E.B. (convener), *Course on data management and related software in geothermal applications.* World Geothermal Congress 1995, IGA pre-congress course, Pisa, Italy, May 1995, 117-130.

Bertani, R., 1995: Data base System. In: Anderson, E.B. (convener), *Course on data management and related software in geothermal applications.* World Geothermal Congress 1995, IGA pre-congress course, Pisa, Italy, May 1995, 85-106.

Bolsky, M., and Korn, D., 1995: *The new Korn shell. Command and programming language.* Prentice Hall PTR, USA, 400 pp.

Hewlett Packard, 1994: *Installing and administering NFS services.* Hewlett Packard Co., USA.

Montalvo L., F.E., 1996: Tracer modelling and heating mining calculations for the Ahuachapán geothermal field, El Salvador C.A. Report 2 in: *Geothermal Training in Iceland 1996.* UNU G.T.P., Iceland, 1-22.

Monterrosa V., M.E., 1993: *A 3-D natural state modeling and reservoir assessment for the Berlín geothermal field in El Salvador, C.A.* UNU G.T.P., Iceland, report 11, 28 pp.

Mui, L., and Pearce, E., 1993: *X Windows system. Administrator's guide for X11, release 4 and release 5. vol. 8.* O'Reilly & Associates Inc., 346 pp.

Quijano, J., 1994: A revised conceptual model and analysis of production data for the Ahuachapán-Chipilapa Geothermal Field in El Salvador. Report 10 in: *Geothermal Training in Iceland 1996.* UNU G.T.P., Iceland, 237-266.

Rogue Wave Software, 1994: *DBtools.h++. Tutorial and user's guide.* Rogue Wave Software, OR, USA, 192 pp.

Salvania, N., 1995: Development of a geothermal database and resource assesment of Mt. Natib geothermal prospect, Philippines. Report 10 in: *Geothermal Training in Iceland 1996, UNU G.T.P.*, Iceland, 241-268.

Singh, L., Leigh, K., and Zafian, J., 1997: *ORACLE 7.3: Developer's guide.* Sams Publishing, USA, IN, 966 pp.

Van Der Lans, R.F., 1988: *Introduction to SQL.* Addison-Wesley Publishing Co., England, 348 pp.

Wessel, P., and Smith, H., 1991: Free software helps map and display data. *AGU, EOS, 72-41,* 441-446.

Wessel, P., and Smith, H., 1995a: *The Generic Mapping Tools (GMT), version 3. Technical Reference and Cookbook.* School Ocean and Earth Science Technology/National Ocean and Atmospheric Administration, 77 pp.

Wessel, P., and Smith, H., 1995b: *New version of the Generic Mapping Tools released.* EOS American Geophysical Union. electronic supplement, http://www.agu.org/eos_elec/95154e.html.

APPENDIX 1: The queries used to create the database tables.

```
/* Well table  */
create table well (
     well_id       integer         primary key,
     name          varchar2(10)    not null,
     x             number          check (x between 100 and 600000),
     y             number          check (y between 100 and 600000),
     z             number          check (z between 0 and 1000),
     xy_quality    integer,
     z_quality     integer,
     max_depth     number          check (max_depth < 10000),
     area_id       integer         not null
)
storage (initial 10K next 2K pctincrease 0);

comment on table well          is
'Info on wells in El Salvador';
comment on column well.well_id      is
'Identification of the well';
comment on column well.name    is
'Wellname i.e. (AH-1)';
comment on column well.x       is
'X-coordinate(latitude) of the well in mts';
comment on column well.y is
'Y-coordinate(longitude) of the well in mts';
comment on column well.z is
'Z-coordinate(elevation) of the well in mts';
comment on column well.xy_quality    is
'Number indicating the accuracy of location in xy-plane';
comment on column well.z_quality is
'Number indicating the accuracy of location in elevation';
comment on column well.max_depth is
'Is the final depth of the well in meters from wellhead';

/* ---------------------------------------*/
/*      xyquality table      * /
create table xyquality (
     xy_quality          integer         primary key,
     error_num           number(3),
     qual_descrip        varchar2(20)    not null
)
storage (initial 10K next 2K pctincrease 0);

comment on table xyquality is
'Quality of altitude and longitude';
comment on column xyquality.xy_quality is
'Number indicating the accuracy of location in xy-plane';
comment on column xyquality.error_num is
'Uncertainty in [m]';
comment on column xyquality.qual_descrip is
'Description of the quality';

/* ---------------------------------------*/
/* zquality table, quality for z  */
create table zquality (
     z_quality           integer         primary key,
     error_num           number(3),
     qual_descrip        varchar2(20)    not null
)
storage (initial 10K next 2K pctincrease 0);

comment on table zquality is
'Quality of elevation';
```

```
comment on column zquality.z_quality is
'Number indicating the accuracy of location in elevation';
comment on column zquality.error is
'Uncertainty in [m]';
comment on column zquality.qual_descrip is
'Description of the quality';

/*   Casing table  creada     */
create table casing (
     well_id       integer        not null,
     odiameter     number         not null check (od between 20 and 2000),
     top           number,
     bottom        number         not null,
     primary key (well_id,odiameter)
)
storage (initial 10K next 2K pctincrease 0);

comment on table casing is
'Information on the casing in the well';
comment on column casing.well_id is
'Number_Id  of the well';
comment on column casing.odiameter is
'Outside diameter of the casing in [mm]';
comment on column casing.top is
'Top of this Outside diameter [m] from wellhead';
comment on column casing.bottom is
'Bottom of the Outside diameter [m] from wellhead';

/*   Log table */
create table log (
     log_id        integer     primary key,
     well_id       integer     not null,
     logtype       varchar2(1) not null,
     logdate       date        not null,
     inidepth      number,
     findepth      number,
     tool_id       integer,
     probe_id      integer,       --  calibration of probe
     plottext      varchar2(12),
     logger        varchar2(20)
)
storage (initial 10K next 1K pctincrease 0);

comment on table log is
'Data of logs for each well';
comment on column log.log_id is
'Unique number for the measurement';
comment on column log.well_id is
'unique ID of the well';
comment on column log.logtype is
'The type id the measurement T or P, refer to table logtype';
comment on column log.logdate is
'The date of the measurement in format 30-JU1-97';
comment on column log.inidepth is
'The initial depth in the measurement in [m] from wellhead';
comment on column log.findepth is
'The final depth in the measurement in [m] from wellhead';
comment on column log.tool_id is
'Unique number of tool_id refer to table tool';
comment on column log.probe_id is
'Unique number of probe_id refer to table probe';
comment on column log.plottext is
'Text for put in the caption';
comment on column log.logger is
'Name of the logger';
```

```
/*   Series table    */
create table series (
     series_id          integer     not null,
     well_id            integer     not null,
     logtype            varchar2(1) not null,
     start_date         date        not null,
     end_date           date,
     probe_id           integer,
     tool_id            integer,
     tool_depth         integer,
     comments           varchar2(30),
 PRIMARY KEY (series_id)
)
storage (initial 2K next 1K pctincrease 0);

comment on table series is
Informatin on measurements with time in a well';
comment on column series.series_id is
'Number unique for each time-serie';
comment on column series.well_id is
'Number unique identifing the well refer to table well';
comment on column series.logtype is
'Type of measure T, P, V, W, H refer to table logtype';
comment on column series.start_date is
'Date when start the serie of measurement';
comment on column series.end_date is
'Date when finish the serie of measurement';
comment on column series.probe_id is
'Id associated with the name of the probe refer to table probe';
comment on column series.tool_id is
'Id associated with the name of the tool refer to table tool';
comment on column series.tool_depth is
'the depth of the tool';
comment on column series.tool_depth is
'Depth of the tool';


/* -------------------------------------*/
/*        Logtype table        */
create table logtype (
     logtype        varchar2(1)    primary key,
     logdescription varchar2(20)   not null,
     in_short       varchar2(8)    not null
)
storage (initial 2K next 1K pctincrease 0);

comment on table logtype is
'What kind of measurements temp and press';
comment on column logtype.logytype is
'The type of measurement T or P';
comment on column logtype.longdescription is
'Description for each measurement';
comment on column logtype.in_short is
'Description for each measurement';


/*        Tool table           */
create table tool (
     tool_id             integer        primary key,
     tooldescription     varchar2(40)   not null
)
storage (initial 2K next 1K pctincrease 0);

comment on table tool is
```

```
'Info about each truck for making the measurement';
comment on column tool.tool_id is
'Tool_id for each truck';
comment on column tool.description is
'Description for each tool for making the measurements temp and press';


/*        Probe table              */
create table probe (
     probe_id             integer         primary key,
     calibrate_date       date,
     probedescription     varchar2(40)    not null
)
storage (initial 2K next 1K pctincrease 0);


comment on table probe is
'Info about each tool or element to make the temp and press measurements';
comment on column probe.probe_id is
'Probe_Id for each probe or tool';
comment on column probe.calibrate_date is
'Last date when it was calibrate';
comment on column probe.probedescription is
'Description for each tool for making the measurements, calibration';

/* --------------------------------------*/
/*   Logging data table  */
create table logdata (
     log_id          integer         not null,
     depth           number(5,1)     not null,
     depthvalue      number(7,2)     not null,
   primary key (log_id,depth)
)
storage (initial 200K next 10K pctincrease 0);

comment on table logdata is
'Values of measurements with depth';
comment on column logdata.log_id is
'Unique number for the measurement';
comment on column log.depth is
'Depth for each measurement of the well in [m] from wellhead';
comment on column log.depthvalue is
'Value for each measurement of the depth';


/*   seriedata table,    for time data      */
create table seriesdata (
     series_id    integer      not null,
     measuretime  date         not null,
     timevalue    number       not null,
  primary key (series_id,measuretime)
)
storage (initial 200K next 10K pctincrease 0);

comment on table seriesdata is
'Values of measurements with time';
comment on column seriesdata.series_id is
'Unique number to identify a series';
comment on column seriesdata.measuretime is
'Time when measurements was done';
comment on column seriesdata.timevalue is
'Value of the measurement';
```

APPENDIX 2: A manual page for the shellscript goragraf

```
goragraf  - Program to plot temperature and pressure measurements
            either by depth or elevation. By default, goragraf plots
            all the measurements.

Usage: goragraf [options]

flags: (order in not important)                                  {defaults}
  -w number   : Log_id of the well to plot the data                    { }
  -P          : Plot the Pressure (P) instead of the temperature       {T}
  -L          : List the data of the well                              { }
  -e          : Plot the graph in elevation                        {depth}
  -H text     : Header in the graphic  separated by (/)
                      {Ahuachapan/Temperature Profiles/Well AH-14}

  -v list     : Select from the list that is gotten using
                the flag -L before  (e.g: -v '1 3 5 6')              {all}
  -l          : Plot in colour                               {Black&White}
  -a number   : Type of measurements
                (0:all, 1:Amerada, 2:GO)                               {0}
  -b min max : Min/Max for T/P-exis          (x-axis)         {defaults}
  -c min max : Min/Max for logs (depths)    (y-axis)          {defaults}
  -B t1 t2    : tick-marks for      T/P-axis                  {defaults}
  -C t1 t2    : tick-marks for   depths-axis                  {defaults}
  -X length   : Length (cm) T/P-axis          (x-axis)              {10}
  -Y length   : Length (cm) depths-axis       (y-axis)              {15}
  -t size     : Size of symbol syze in the profiles (cm)          {0.15}
  -i dist     : Distance between legend texts(cm)                  {0.5}
  -p size     : How to plot data                                      {1}
                size = 0 --> plot all data points
                size > 0 --> distance between symbols in (cm)
                the following flags are the logo

  -fl font size : Font and size for GMT-graphic
                              {11 10 ;(Courier-BoldOblique, 10 size)}
  -fh size    : Header text size                                     {14}
  -fa size    : Axis text size                                       {14}
  -fo s1 s2   : Change the parameters with the logo depends on -OS flag
                  -OS 1 --> logo size (cm) and font size       {0.75 14}
                  -OS o --> logo size (cm) and font size          {17 10}
                  -OS e --> font and text size (cm)               {0 10}
  -OS type    : Type of logo OS, with two lines text                  {1}
                (l = logo; o = OS; e =  OS text; s = no logo)
  -o x-coord y-coord  : Where to put the logo (cm)                    {}
  -O txt1 txt2   : Text with OS-logo           {(day) (logname well_id)}
  -u          : No legend in the graph                                { }
  -r          : No frame around graph                                 { }
  -h          : Help

Examples:
      goragraf -h              (Help)
      goragraf -w 1014         (All profile temperature of AH-14)
      goragraf -w 1014 -L      (List all temperature data of AH-14 well)
      goragraf -w 1014 -P      (All pressure  data of AH-14 well)
      goragraf -w 1014 -P -L   (List all temperature data of AH-14 well)
      goragraf -w 1014 -H 'Ahuachapan/Temperature Profiles/Well AH-14'
      goragraf -w 1014 -P -v '1 3 5 6'

Author: Ricardo Ventura - October 1997,
The first version was by Grimur Bjornsson & Thordur Arason
and it was modified by Sigvaldi Thordarson - July 1997
```