

# New database for nighttime programs at the National Optical Astronomy Observatories

C. A. Pilachowski<sup>a</sup> and C. Brown<sup>a</sup>

National Optical Astronomy Observatories<sup>b</sup>, PO Box 26732, Tucson, AZ 85726

## ABSTRACT

The National Optical Astronomy Observatories has developed a new database system, ALPS, to track proposals for telescope time from original receipt through the review process, scheduling, observing, and final statistical reporting. The database is written in Microsoft Access, and is integrated with observatory operations. Proposals arrive in a  $\LaTeX$  format and are parsed into files suitable for import into Access using a Perl script running under Unix. The database system provides tools to support all activities associated with handling proposals, including support for the Telescope Allocation Committee through reviewer assignments, grades imported via the Web, and comments for the principal investigator. The telescope schedules are prepared through a scheduling interface, and the final schedule is posted automatically to the Web. Statistics on telescope usage (hours of observing, time lost) are collected via the Web and imported into the database as well.

The new database has been in operation since March, 1997, for proposals submitted for observing time at the Kitt Peak National Observatory, and has been installed at Cerro Tololo Interamerican Observatory as well. The program is written to be easily adaptable for new facilities which will be available through NOAO, including public access to time at independent observatories and access to the Gemini telescopes.

**Keywords:** observing proposals, database, telescope allocation

## 1. INTRODUCTION

The Gemini 8-m telescopes will begin scientific operations in the years 2000 (for Gemini North) and 2001 (for Gemini South). The National Optical Astronomy Observatories (NOAO) will accept proposals for observing time within the U.S. allocation of time on these telescopes starting in 1999. As the Gemini telescopes ramp up, and as NOAO also accepts proposals for community access time on the Hobby-Eberly Telescope and the MMT, we anticipate that the number of telescope proposals received and processed by NOAO will increase dramatically. To prepare for these changes, NOAO is developing a new proposal database (ALPS - Addresses, Lists, Proposals and Schedules) and procedures to accommodate an increasing number of observing proposals for a mixture of sites and instruments.

## 2. REQUIREMENTS

Our task is to provide a modern database program for supporting telescope proposals and observatory operations which is easy to use and to maintain, which provides for flexibility to accommodate the changing needs of NOAO, which meets the needs of the NOAO sites for operations and information, and which is compatible with adopted NOAO/Tucson standards for PC software. To achieve these goals, we undertook to develop a new database program to replace software in use at NOAO since 1981.

NOAO receives nearly a thousand proposals annually for its nighttime programs, and that number is expected to increase as new facilities come on line. Wherever possible, we have tried to facilitate automatic entry of data into ALPS, such as proposal data, reviewers' grades and comments, observing run preparation data, observing statistics, observing run evaluations, etc. Entries are reviewed during the import process for accuracy to assure the quality of the data. Where manual entry of data is required, special attention has been paid to facilitate ease of use.

Because of the changing nature of NOAO's program and the facilities we support, ALPS must be easily maintained and flexible in accommodating changes. The specific telescopes and instruments offered to the community will

---

<sup>a</sup> Authors' email: Pilachowski – cpilachowski@noao.edu; Brown – cbrown@noao.edu.

<sup>b</sup>The National Optical Astronomy Observatories are operated by the Association of Universities for Research in Astronomy, Inc. (AURA), under cooperative agreement with the National Science Foundation

change rapidly during the next 5 years. Such information must be easily updated each semester. Procedures will be evolving as well. For example, NOAO's Galactic and Extragalactic Telescope Allocation Committees (TAC) must be augmented by additional discipline panels as the number of proposals increases. Each site also has special requirements which must be accommodated in the database

In addition to supporting the telescope allocation process, the database must also be available to observatory staff to support functions associated with observatory operations. These include access to information about scheduled programs and observing needs, telescope logs, and other records. Access to the ALPS database is restricted to staff who are directly involved in the proposal, allocation, and usage statistics. Broader access to information in the database is to be provided through Web pages.

ALPS++ is designed to incorporate older data from 1981 to the present from an earlier, UNIX-based database to maintain access to the information.

### 3. IMPLEMENTATION

The NOAO administrative staff uses the Microsoft Office 97 Professional \* software suite, an integrated group of programs (Word, Excel, Access, PowerPoint, and Schedule+) that allows the user to perform word processing, spreadsheet, database management, presentation graphic, and group scheduling tasks. For compatibility with these operations, we selected Microsoft Access as the development environment and database management product for creating ALPS.

In loose terms, ALPS is simultaneously referred to as a "database," an "application," and a "program." It is important, however, to clarify these terms. A database alone is just a collection of related data. It is the basis for a database application, which is a programmed system that is designed for two main purposes:

- Data input/collection and storage, which involves adding, deleting, and updating the data in the database.
- Data output and analysis, which involves providing various ways to view, on screen and in print, the data in the database.

Technically, the ALPS application is a single file that contains six types of objects used to manage the data in the ALPS database:

- Tables are the fundamental, two-dimensional structures that store the data being managed by organizing it into columns and rows. Each row in a database table is a single record.
- Queries are stored Structured Query Language (SQL) commands that are used to retrieve a specified set of data. Typically, the data is combined from two or more tables in a particularly meaningful way. SQL is the most popular non-procedural data access language today on computers of all sizes. Because it is supported by hundreds of databases on platforms ranging from billion-dollar supercomputers down to thousand-dollar personal computers, it is the /it de facto data access language standard.
- Forms are what the user sees – they display data more visually and allow the user more control over the data being presented than is possible using a simple, two-dimensional, spreadsheet-style table.
- Reports are the templates that allow the user to print data in a useful format.
- Macros are simple stored sets of instructions that automate tasks that need to be performed on data. No macros are used in ALPS, however, because they don't allow the precise control over actions that can be obtained by using Visual Basic for Applications (VBA) programming instead.
- Modules contain customized, application-specific VBA code that has been programmed to respond to events initiated by the user, such as clicking a button on a form.

---

\*Microsoft, Access, Visual Basic for Applications and Windows are either registered trademarks or trademarks of Microsoft in the United States and/or other countries.

ALPS represents a real-world system that has been modeled in a relational database application. The relational database model was conceived in 1969 by E. F. Codd<sup>1</sup>, then a researcher at IBM. The model is based on mathematical theory or, more specifically, on the disciplines of set theory and predicate logic. The basic idea behind the relational model is that a database consists of a series of unordered tables, or relations, that can be manipulated using nonprocedural operations that return tables. This model was in vast contrast to the more traditional database theories of the time, which were more complicated, less flexible, and depended on the physical methods used to store the data.

The process of modeling NOAO’s proposal-management system in a relational database application began with deciding which tables to create, which columns they would contain, and the relationships between the tables. The benefits of a database that has been designed according to the relational model are numerous:

- Data entry, updates, and deletions are efficient.
- Data retrieval, summarization, and reporting are efficient.
- Because the database follows a well-formulated model, it behaves predictably.
- Because much of the information is stored in the database rather than in the application, the database is somewhat self-documenting.
- Changes to the database schema (table definitions) are easy to make.

Tables in the relational model are used to represent “things” in the real world. Each table should represent only one type of thing. The things (or entities) can be real-world objects or events. At NOAO, the real-world objects we sought to model in the database included proposals, investigators, telescopes, instruments, and telescope allocation committees. Events that we sought to model included the proposal grading process, the process of scheduling observing programs, and observing runs themselves.

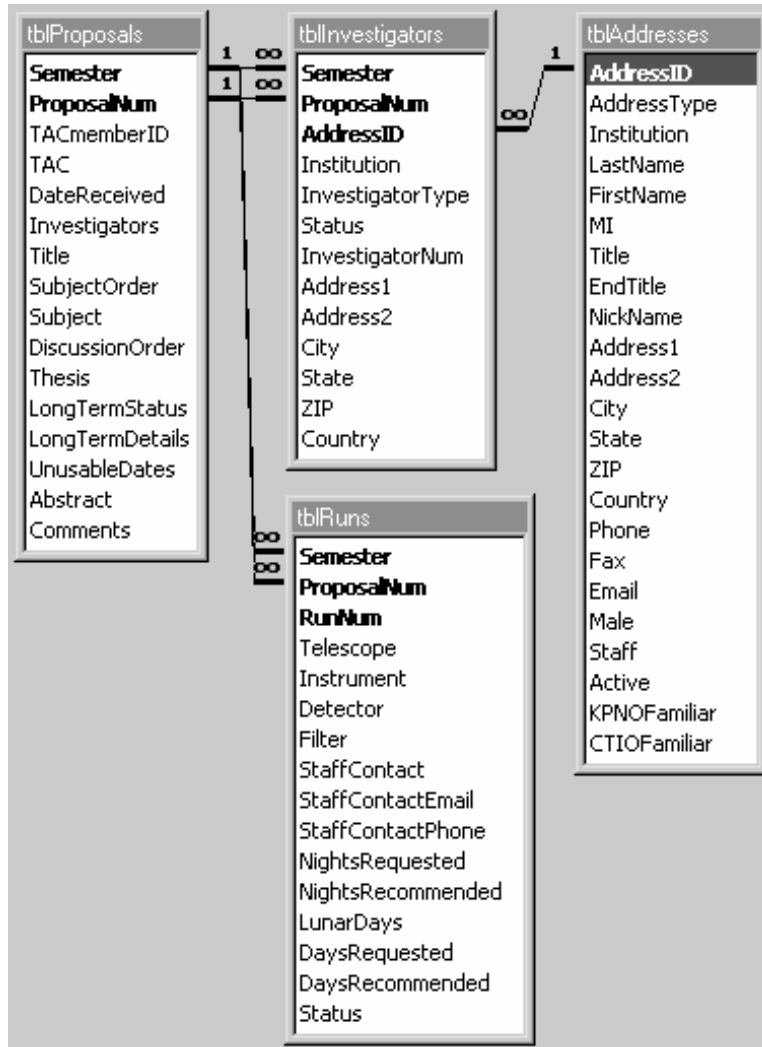
Tables consist of rows and columns. The relational model dictates that each row in a table must be unique. If duplicate rows are introduced into a table, a given row cannot be addressed uniquely via a program, leading to ambiguities and other problems.

Referring to Figure 1, the symbols 1 and  $\infty$  represent the type of relationship between the corresponding tables. (The symbol “ $\infty$ ” is read “many.”) Relationships can be classified into three types: one-to-one, one-to-many, and many-to-many.

One-to-one relationships, where each entity on each side is related to at most one entity on the other side of the relationship, are rare in database design, and none exist in ALPS. This type of relationship is often created to get around some limitation of the database management software rather than to model a real-world situation. For example, a one-to-one relationship might be necessary because of security or performance concerns or the limit of 255 columns per table.

Two tables are related in a one-to-many relationship if, for each row in the first table, there can be zero, one, or many rows in the second table. For each row in the second table, however, there is only exactly one row in the first table. All of the table relationships in ALPS are one-to-many relationships. The single side of the relationship is denoted by the symbol 1, and the many side of the relationship is denoted by the symbol  $\infty$ . For instance, in the relationship from tblProposals to tblRuns, one proposal may have more than one observing run, but an observing run is proposed on at most one proposal (or so we will assume).

Two tables have a many-to-many relationship when, for each row in the first table, there can be many rows in the second table, and for each row in the second table, there can be many rows in the first table. Many-to-many relationships can’t be directly modeled in relational database programs, including ALPS. These types of relationships must be broken into multiple one-to-many relationships. For example, a proposal may have more than one investigator, and an investigator may be on more than one proposal. Thus, the tblProposals table in ALPS is actually related to the tblAddresses table using a many-to-many relationship. To model the relationship between these two entities, a third table was created, a linking table, called tblInvestigators, which contains a row for each proposal-and-investigator combination, or transaction.



**Figure 1.** The relationship between the tables for investigators and for proposals.

ALPS is based on the object model for the manipulation of data. The object model it employs is called Data Access Objects (DAO), a shared component of Microsoft Office. By manipulating DAO programmatically using VBA, an event-driven programming language, a programmer can manipulate not only the data in an Access database but the data in a Word document, Excel spreadsheet, or any other Office program. Using DAO and VBA, Office products can “talk” to one another. For example, ALPS merges name and address information, reviewer comments, and scheduled observing program dates and information, all collected in the ALPS database, with the appropriate letter created by the user using Microsoft Word. Depending on criteria stored in ALPS, ALPS can determine whether or not a proposed observing program was granted time, and subsequently produces either a “yes” or a “no” letter – whichever is appropriate. The user retains the flexibility of using Word to create a formatted letter, since Word is really a more appropriate tool for this particular task than Access. The user can change the wording and formatting of the letter at will, as long as the placeholders for the merged data from ALPS are left undisturbed.

ALPS, as with any Access database application, can be used in three different types of environments: standalone, file-server, and client-server:

- In a standalone environment, the database application file simply resides on one computer’s hard drive. In this type of environment, the database can be accessed and the application can be used by only one person at a time. Furthermore, any person wishing to access the database must physically relocate to the computer’s site.

The opposite of a standalone environment is loosely referred to as a client-server environment. In its most basic sense, a client-server application is one that divides the execution of tasks between two or more separate processes (running programs). A server is any process that offers services to other processes (the clients). These processes are typically located on different computers connected by some sort of network, but they need not be. There are two types of client-server environments:

- In a file-server environment, the database application file is simply moved to a network server's hard drive. In this case, no active components at all are running on the server. All the data processing, or "work," is done by the client workstation retrieving the database application across the network. When it is finished, it writes the result of its work back to the server. All of the client's processing must be done in memory.
- In a true client-server environment, an Access application is used only to provide an interface, or front-end, to data residing on a dedicated SQL-based database management server (the back end of this arrangement). The burden of data storage and manipulation is shifted to the server; individual workstations and copies of the Access application running on them serve merely as windows to the processes occurring on the SQL server.

The standalone environment was ruled out as an inefficient arrangement, since multi-user access as well as the convenience of using ALPS both from one's own desktop as well as simultaneously with other Windows applications was desired. The choice then came down to whether ALPS should be implemented in a file-server or a client-server environment. We selected the file-server environment since it is more compatible with existing hardware configurations and offers some advantages in performance over the client-server environment.

The database is expected to grow by about 2 MB each semester, or 4 MB per year. An Access database can hold up to 1.2 GB of data and can handle 50 concurrent users. Should the need arise to migrate to a larger-capacity client-server environment, Microsoft Access does offer upsizing tools to accomplish this task.

If ALPS is to be operated in a file-server environment, however, attention must be paid to the hardware on which the application will be run. The minimum system requirements for running ALPS are:

- A personal computer with a Pentium 60-MHz or higher microprocessor (Pentium 133 or higher recommended).
- Microsoft Windows 95 or Windows NT Workstation operating system version 4.0 or later.
- 32 MB of RAM.
- Minimum 25-55 MB of available hard disk space.
- Microsoft Mouse or compatible pointing device.

As discussed above, Access puts all of its objects by default into a single database file. Performance can suffer considerably under this scenario, however, because every time an object (for example, a form) is used, it must be sent across the network to the user. In a production setting, in which only the data itself is being updated, rather than the object being used (the form), much of this network traffic is unnecessary.

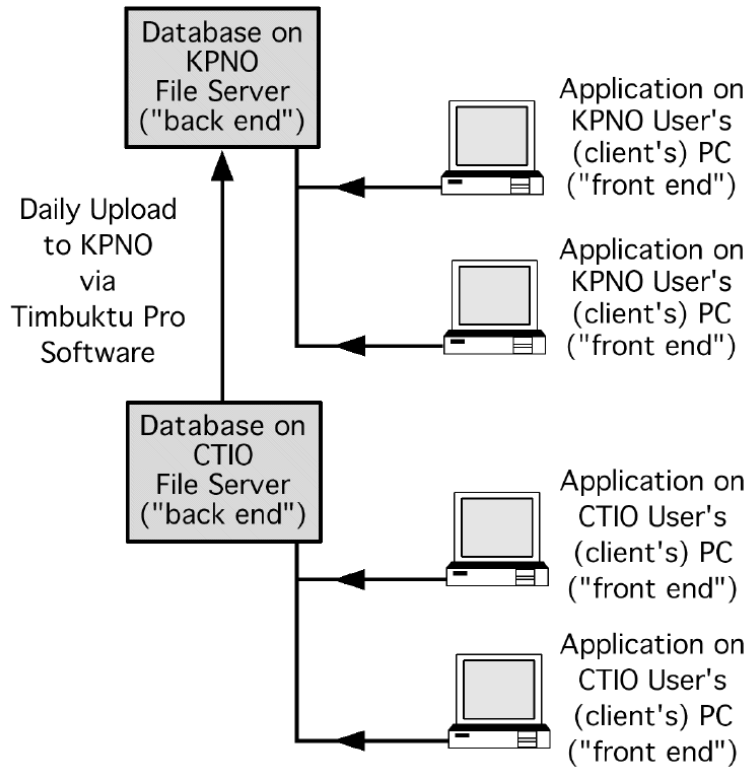
To eliminate this unnecessary traffic, the ALPS database was split into two files: a "data" database file containing the application's tables and a "program" or "application" file containing all other elements of the application (the forms, reports, queries, and modules). The data file is installed on the file server and a copy of the application file is installed on each client workstation. As in a true client-server relationship, the data file can also be referred to as the back-end database and the application file can be referred to as the front-end database. Figure 2 illustrates the relationship between the file server and each client application.

This two-tier architecture improves the way ALPS is developed, deployed, and maintained. The advantages of this approach are:

- Performance (especially user-interface performance) is improved considerably.
- Temporary tables used to import data can be created on each workstation, eliminating possible naming and locking conflicts between multiple users of the same process or object at the same time. For example, using this approach allows more than one user to import proposals at the same time.

- Splitting the database makes updating the application easier because the data and application are kept separate. Changes to the application can be made and easily merged back into the application database without disturbing the data.

### Interoperability between Servers and Applications



**Figure 2.** The relationship between the file server and each client application.

The application file is programmed to link with, or “look at,” the tables in the data file residing on the central server. When linking tables from a back-end database, it’s desirable to use the Universal Naming Convention (UNC) path (for example, \\ServerName\ShareName\Data.mdb) instead of a mapped drive letter (such as g:\Sharename\Data.mdb). This way, client workstations using the front-end database aren’t required to use the same network drive letter configurations.

Splitting the database also allowed us to maintain two separate sets of data in geographically distant locations: Kitt Peak National Observatory (KPNO) in the United States and Cerro Tololo Interamerican Observatory (CTIO) in Chile. We chose to maintain two separate data databases primarily for performance reasons: it made more sense for CTIO to be able to access ALPS on its own local file server. The trade-off of allowing CTIO users speedy access to ALPS, however, presented us with a remote administration problem, which we solved by utilizing Farallon’s Timbuktu Pro for Networks<sup>†</sup> software. Timbuktu Pro allows the database administrator to log on to any CTIO computer on which Timbuktu Pro is running, including both the CTIO ALPS file server and any CTIO ALPS workstations. Using Timbuktu Pro, the database administrator can:

- Support CTIO end users from the United States. Using remote control, the database administrator can observe user problems, take control, and solve them.

<sup>†</sup>Timbuktu Pro for Windows is a registered trademark of Farallon Communications, Inc.

- Upload application updates to each CTIO workstation running ALPS.
- Download CTIO data to KPNO.

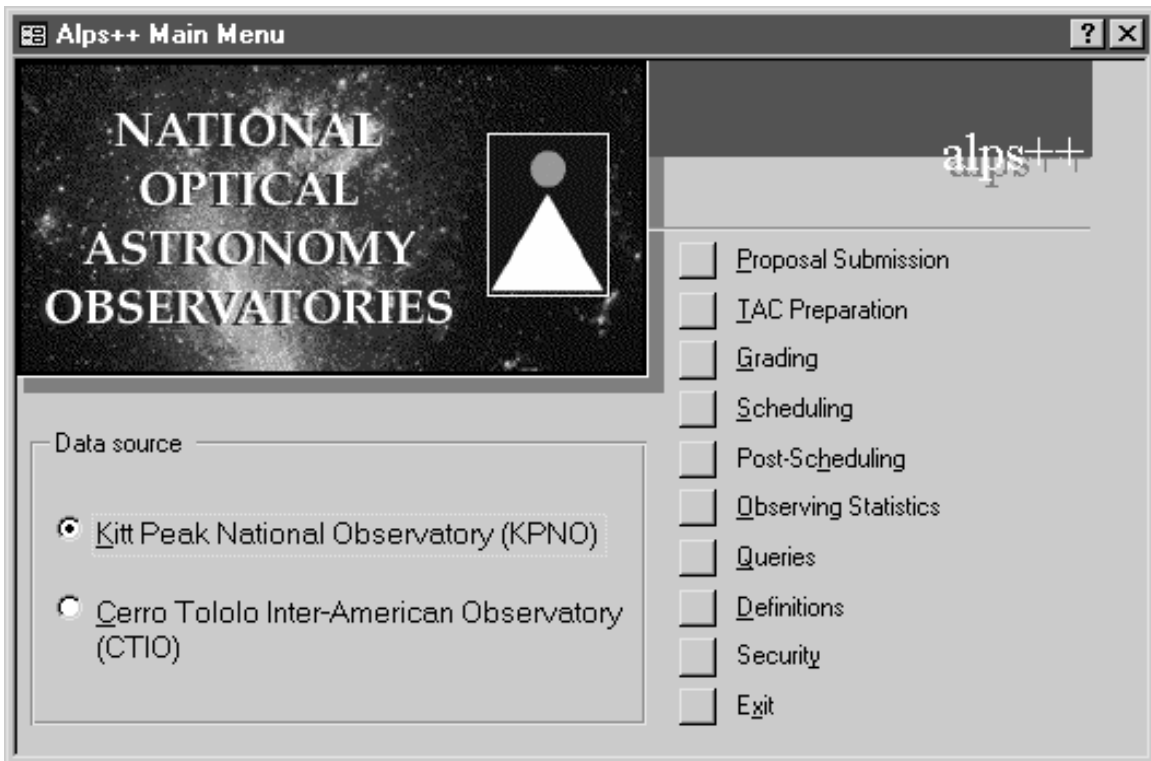
This third use of Timbuktu Pro became necessary for the same reason we decided to maintain two separate, local copies of ALPS in the first place. Although Timbuktu Pro allows us to maintain the database easily from a remote location, it is not feasible for accessing CTIO's data quickly online. For the same performance reasons we utilized a local file server for ALPS at CTIO, we likewise needed to access CTIO data locally at NOAO headquarters in the United States. Not only do we need to query CTIO's data and obtain output in the form of reports, but we actually need to enter and edit CTIO data during CTIO's Telescope Allocation Committee meeting in the United States.

Finally, we note that the ALPS server shares a cross-mounted disk with a Unix Sun workstation to allow easy file transfers between the two systems involved with NOAO observing proposals.

#### 4. THE USER INTERFACE

Users enter the database through the main ALPS menu. Access is password controlled, and is limited to the administrative and technical staff with a specific need to enter or access data.

The main menu, shown in Figure 3, is organized by function. The principal functional tasks of ALPS include the importation of proposals; the preparation of materials for the TAC, the support of the TAC meeting, the preparation of the telescope schedules, the communication with investigators about the outcome of their proposals, and the collection and reporting of statistics about telescope usage required by the funding agency. In addition, the ALPS database includes a Master Address File, which includes past and current investigators, institutions, and other correspondents of NOAO.



**Figure 3.** The main menu of the ALPS database program.

Available within each main menu item are the forms needed to carry out the essential tasks of that function. For example, within the area of proposal importation are forms for importing proposal files, reviewing and editing investigator and observing run information, updating tables of available telescopes and instruments. Tasks associated

with the preparation of TAC materials include the assignment of lead reviewers, the sorting of proposals to particular TAC's, etc. Tasks associated with the support of the TAC meeting include the importation of reviewers' grades, the editing of grades during the TAC meeting and the entry of recommendations on time allocations, and the importation of TAC comments after the meeting. Forms are designed to assist novice users to understand their use easily, and accelerators are provided to assist experienced users who work quickly.

In addition to the forms needed to carry out database tasks are the reports (i.e. printed summaries) which must be generated at each stage. These include lists of proposals in numerical order and in alphabetical order by principal investigator or sorted by telescope, competition lists for the TAC, lead reviewer assignments, lists of thesis programs, TAC and investigator correspondence, scheduling lists, telescope schedules, etc. Some reports are issued in ASCII form to facilitate posting to the World Wide Web (e.g. TAC grade entry forms and telescope schedules) and distribution by email and for editing on Unix computers.

## 5. TELESCOPE PROPOSALS

Among the most important tasks of the ALPS database is the support of the review process for proposals for telescope time at facilities available through NOAO. Telescope proposals are received semi-annually in March and September. The proposals are submitted by email using a  $\LaTeX$  template and .sty file or over the Web<sup>2,3</sup>. The use of a  $\LaTeX$  template (Web proposals are also incorporated into  $\LaTeX$  templates) is mandated to provide to our user community and reviewers a clear, readable, printed copy of the proposal which can include special symbols, equations, figures, and tables. Once the proposal has been received, logged, and corrected for any  $\LaTeX$  errors, a Perl CGI script<sup>4</sup> is used to interpret and extract the necessary information into a text file accessible to the Access database.

Proposals are imported into the database as they become available. Information imported into the database includes the names and addresses of the investigators, including email, fax, and telephone, the title and abstract of the proposal, a subject designation, the telescope, instrument, and scheduling requirements, and any special status (e.g. thesis programs, long-term programs). At import, the information in the proposal is checked for completeness and accuracy and cross-referenced to the master address file of NOAO users. Instrument and telescope requests are cross-checked against tables of the available facilities. Inadequacies in the data are corrected in the database and/or in the original  $\LaTeX$  file. Data problems include such items as a missing field, an unknown telescope or instrument designation, or an ambiguity in the requested observing run parameters. This quality check of the proposal information requires a few minutes per proposal, but is essential to maintain the veracity of the data.

## 6. SUPPORT OF THE TELESCOPE ALLOCATION COMMITTEE

Once the proposal information is imported, the database can be used to support all the necessary functions of the allocation process. The database includes assignment to a particular Telescope Allocation Committee, the assignment of a lead reviewer to each proposal, and the generation of the necessary documentation to support the TAC process, including proposal and competition lists, lead reviewer assignment sheets, reviewer rating forms, proposal status reports, etc. Necessary materials are mailed to the Allocation Committee members and distributed in-house. ASCII lists of proposals are generated and distributed to TAC members electronically to facilitate the preparation of comments to be returned to the principal investigators.

Information from the ALPS database is used to generate (again via Perl CGI scripts) passworded Web pages for grade entry by TAC members. TAC members have the opportunity to review and edit their grades during multiple sessions prior to submission. Once grades are submitted via the Web, a Perl CGI script creates a grade import file so that each TAC member's grades can be imported into the database. After all grades are entered, preliminary competition lists are printed, ranking proposals by their average grade. Identifying flags note proposals for which the dispersion in grades is large or for which a TAC member has a conflict of interest. The TAC Chair receives a special list which includes the individual grades and the lead reviewer assignment for each proposal, in the order that proposals will be discussed.

During TAC meeting, the database is used to record grade changes and recommendations on the allocation of nights by the TAC. Updated competition lists are reprinted as needed for use by the TAC.

Special attention has been given to improving the process for writing and sending out TAC comments on each proposals. The TAC Lead Reviewer is responsible for writing comments which reflect the discussion of the proposal

by the TAC. Laptop computers are provided to TAC members for writing comments. These ASCII files are then imported into the database and become part of the proposal record. Comments are edited as needed before being sent out to the Principal Investigator.

## 7. THE TELESCOPE SCHEDULE

After the TAC meeting, the observatory staff use the ALPS database to write the telescope schedules. New competition lists are generated which merge the recommendations of the Galactic and Extragalactic TAC's, and which include scheduling information for use in writing the schedule. Tools are provided to enter schedule data, and the ALPS database can print preliminary and final versions of the schedule. The database includes information about holidays, phases of the moon, etc., for inclusion on the schedule.

When the schedule is complete, and after any subsequent schedule change, an ASCII version of the schedule is exported via email, and another Perl CGI script is used to post the schedule automatically to the Web.

The database is then used to prepare information to send to principal investigators about the outcome of their proposals. The mailing includes a letter awarding or denying time, and a proposal status report which includes TAC comments as well as observing run information if the proposal is successful. Staff contact information particular for each observing run is also included.

## 8. OBSERVATORY OPERATIONS

The ALPS database ties together numerous functions in support of observatory operations. Support staff are provided information about scheduled observing runs and investigator contact information. Database searches provided needed answers to ad hoc queries (e.g. a list of investigators using a particular instrument or a list of thesis programs which received telescope time).

Statistical information on telescope usage is also gathered into the database. Each night, telescope operators (and observers at the smaller telescopes) enter via the Web the hours the telescope was used and the hours lost due to weather, equipment failure, or engineering or maintenance. When the Web page is called from a telescope computer, the data and telescope as well as information from the database including the scheduled observing program, the observers' names, and instrument are automatically filled in. (If a change has occurred in the scheduled program, these defaults can be replaced with corrected information.) Again, a Perl CGI script is used to create an import file to load the observing log information into the database. The information can then be extracted to provide the necessary statistical reports on the operation of the observatory required by funding agencies.

Future plans include the integration of pre-observing run information such as the names of investigators coming to the observatory, their housing and travel requirements, and any special instrumentation or data reduction needs. We also expect to include observing run evaluations into the database.

## 9. EXPERIENCE TO DATE

While the database is still under development, ALPS has been used successfully for two semesters at the Kitt Peak National Observatory, and is in use as well at the Cerro Tololo Interamerican Observatory. Response to the new program has been positive, with requests for new features to augment its capabilities far outstripping the available programming resources. The database has reduced substantially the workload involved in supporting the proposal process and observatory operations. During a recent semester, for example, all KPNO proposals had been imported into the database at the end of the day following the proposal deadline. We are confident that the new process will permit us to handle the anticipated increase in the number of proposals submitted to NOAO's nighttime programs.

During the September, 1997, proposal round, the ALPS database was used for the first time at the Cerro Tololo Interamerican Observatory. Proposals were received at NOAO/Tucson and mirrored to Chile, where  $\LaTeX$  errors were identified and corrected, import files were created, and proposals were imported into the database. The CTIO database is copied daily to Tucson and is available to NOAO's Science Operations staff to assist CTIO's allocation process, including CTIO TAC mailings and support. An updated version of the CTIO database, including reviewer comments from the CTIO TAC, was copied to Chile following the TAC meeting to be used to write the telescope schedules. Once completed, the CTIO schedule could be posted immediately to the Web in the same manner as for KPNO.

As with any new system, some frustrations are inevitable. The major difficulties encountered in the first year of ALPS operation include:

- The changeover from the original database programs at KPNO and CTIO has necessarily occurred over a period of 1-2 semesters, requiring effort to maintain duplicate systems. The process is largely complete at KPNO, but is still underway at CTIO.
- Due to inclement weather and other natural disasters, the network connection to the ALPS server in Chile has not been as reliable as we had hoped. The response time for database upgrades, bug fixes, and user support has been affected, leading to some frustration at both sites.
- Users not familiar with the table and relational structure of the database have some difficulty querying its data, and writing SQL queries is neither fun nor intuitive for most lay ALPS users. We hope to address this problem as described below.
- The balance between purity of design and performance has also been a challenge. ALPS is a normalized database. With few exceptions, the manipulation of tables in ALPS doesn't depend on a specific ordering of columns or rows. The process of normalizing a database isn't theoretical: non-normalized databases cost more to build, run and maintain than normalized databases. Querying a non-normalized database costs more and takes longer.

Storing the observing schedule in a purely normal form, however, results in having to wait about three minutes, using the 60- to 166-MHz processors generally available at NOAO, every time a user wants to view or print a cross-tabulated observing schedule. The acceptable compromise seems to be to store a usable schedule in a standard two-dimensional, non-relational, spreadsheet-style format; that is, each telescope would have to be a column in this interim table and each night on the schedule, broken down into the first and second half of the night, would have to be considered a row.

## 10. PLANS FOR THE FUTURE

While ALPS is nearly complete in its initial implementation, we anticipate numerous upgrades and additions to improve the utility and functionality of the database. In addition to many small improvements, we hope to carry out several major upgrades to the database. These include:

- The Web interface. The capability of the database to produce HTML pages for use with the World Wide Web has been significantly enhanced with the recent conversion to Microsoft Office 97. We plan to use this capability to distribute information needed by in-house staff for observatory operations over the Web. Examples include more detailed telescope, instrumentation, and schedule information, and investigators' contact information (e.g. email addresses, FAX numbers, etc.), and nightly or weekly reports of telescope operations from observers' logs.
- *Ad hoc* query interface. While standard forms and reports are available through the database menus, questions often arise which require a special query. At present, a skilled user can respond to such queries, but an *ad hoc* query interface is needed to allow all users to access information in the database. Examples of such queries include a list of observers from a particular institution, a list of observing runs with a particular instrument, or a list of thesis programs from 1990-1995.
- Archival data. NOAO maintains historical records in electronic form of observing programs since 1981. These records need to be imported into the database to allow investigation of long term trends and to provide historical information on programs carried out at NOAO. Changes in instrumentation, telescopes, and vocabulary in the past 17 years will make this a challenging project to maintain continuity and consistency in the database.
- Schedule Preparation. Modern artificial intelligence methods will allow preliminary telescope schedules to be drafted in software, accounting for scientific program requirements for season and lunar phase and for the observatory's needs for block scheduling and limitations on weekend instrument changes and observing run starts. We hope to implement such scheduling aids to assist in the preparation of telescope schedules.

## 11. ACKNOWLEDGEMENTS

We are grateful to Jeannette Barnes for her capable assistance in the preparation of this manuscript. Jeannette Barnes and David Bell also designed and implemented the various Perl CGI scripts and Web interfaces which are used so extensively to access and display data and information with the ALPS database. We are very grateful for their assistance. We also thank the users of ALPS at NOAO, KPNO, and CTIO who have helped with testing and who have provided valuable and thoughtful suggestions for improvements to the database.

## 12. REFERENCES

1. Codd, E. F. 1970, "A Relational Model of Data for Large Shared Data Banks," *Communications of the Association for Computing*, **13**, 377-387.
2. Bell, D. J., Barnes, J., and Pilachowski, C. A., "The NOAO Web-based Observing Proposal System," *A.S.P. Conference Series* (in press), 1998 (Astronomical Data Analysis Software and Systems VII).
3. Pilachowski, C. A., Barnes, J., and Bell, D. J., "Observing Proposals on the Web at the National Optical Astronomy Observatories," 1998, *Proc. SPIE* **3349** (in press).
4. Bell, D. J., "Filtering KPNO  $\LaTeX$  Observing Proposals with Perl," *A.S.P. Conference Series*, **125**, 371, 1997 (Astronomical Data Analysis Software and Systems VI).