Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

A Generalized Mosaic-to-SQL Interface with Extensions to Distributed Archives

F. Pasian and R. Smareglia

Osservatorio Astronomico di Trieste, Via G B Tiepolo 11, Trieste, I 34131 Italy

Abstract. A graphical user interface based on Mosaic, and allowing form-based access to SQL-driven relational databases, is described. The system is general, due to some simple naming conventions for database tables. This allows a description of the database structure (contents of tables, joins, and views) to be stored in the database itself, and the Mosaic pages to be prepared on-the-fly, with no need for external definition files.

Furthermore, the interface allows X-Y plotting of any two numerical quantities retrieved from the database, and access to data archives distributed over Internet, following the WWW concept. Quick-look data can be displayed, and observational data can be retrieved in their original format (FITS) inside compressed `tar` files. All of these operations are performed on the archive hosts by proper server processes, and no additional software, besides Mosaic, is required at the user's site.

The functionality of the system has been tested on a small archive of objective prism data (images and extracted spectra) and will be implemented for a distributed archive of ISM data.

1. Introduction

The World-Wide-Web (WWW) and NCSA Mosaic can be considered a *de facto* standard for information retrieval, and their wide acceptance is leading to a different and more homogeneous design of interfaces to archives and databases in astronomy. The advantages of using Mosaic as an interface are numerous: distribution and development of the software are taken care of at NCSA, the Mosaic client handles all the intricacies of the man-machine interface, the development rules are easy to learn, and data centers do not need to bother about distributing and maintaining user interface or commercial software at a number of external sites.

The information providers can therefore concentrate on building servers able to guarantee efficient access to their data, without the need of building *ad-hoc* interfaces to their systems. It is essential that data access tools are flexibly built and can easily communicate with this *standard* user interface. These are the basic concepts we followed while implementing the system described in this work.

2. System Architecture

The user interface allows access to the underlying structure of the archive: a catalogue managed by a commercial SQL-based database management system, and a data bank holding the original data and compressed quick-look data.

The interface itself, which is based on NCSA Mosaic, allows a number of operations on the archive: selection of specific database tables or views to be accessed; browsing of the catalogue (database tables or views) by specifying query constraints inside forms to be filled by the user; possible editing of the resulting SQL query; display of the results in a HTML page; graphical visualization of the numerical results derived from a query; display of quick-look 2-D images and finding charts; display of quick-look 1-D graphs (e.g., extracted spectra); and retrieval of observational data from the data bank.

The structure of the system is shown in Figure 1. Users access the archive through a Mosaic interface; at the host node, the http messages coming from the interface are interpreted by a **Protocol Handler**, and different specialized servers can be started:

- The **SQL Server** builds a SQL query from information received from the interface, and passes back to the interface the result of the query or quick-look data (if any).
  - The **DBMS Server** extracts from the database information on the table or view to be accessed and builds the Mosaic interface page; in other cases passes the SQL query on to the DBMS and gets results.
- If the data are stored locally, the **Data Server** performs a username/password check through the DBMS Server, accesses on-line databank files and sends them to the user, or sends retrieval requests for off-line data to the archive operator.
- The **Graphic Server** plots the numerical results of a database query (or of archive data, in the case that quick-look data do not exist), and passes the graphic files back to the interface.

In keeping with the concept of the WWW, this system allows access to archives of data distributed over the Internet. From a logical point of view, it makes no difference if the archive is local or remote: the pointers to the data are stored in the database and, once accessed, are fed back to the interface for standard WWW access. If necessary, the Protocol Handler sends a message asking the remote archive to perform some specific action which may be required (e.g., storing files to be transferred in a specific world-readable directory, running a dedicated process to perform format conversion on archive data, or packing data in a compressed `tar` file).

3. Database Structure and Naming Conventions

The SQL Server uses *standard* SQL calls to access the database in retrieve mode, therefore guaranteeing independence from the SQL dialect used by the specific DBMS. It is, however, another feature that allows the system to be fully

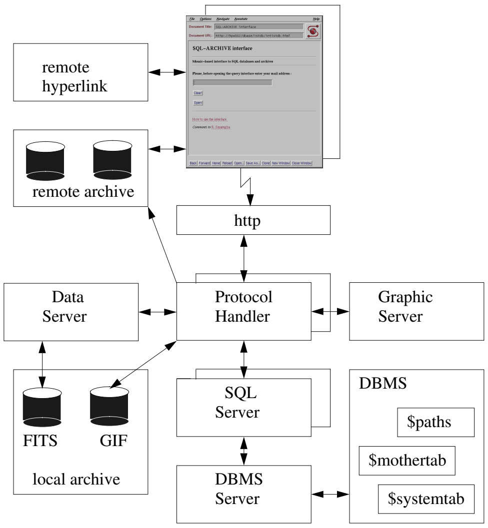

Figure 1. Structure of the system, based on a number of different servers, and allowing access to remote archives and hypertext media via the HTML protocol. The possibility of sending communications to remote archives through the Protocol Handler is shown. In the database, the three special tables allowing system portability are evidenced.

general: all of the interface information is dynamically stored in the database and is retrieved when needed to build on-the-fly Mosaic interface pages.

This mechanism works as follows: when it is necessary (i.e., when a user has issued a request to access data stored in the database), the DBMS server first retrieves from the database information on the table (or view joining a number of different tables) that needs to be accessed. Such information is used to build on-the-fly the form the user will to use to browse the contents of the database. The form, in HTML format, is then fed to the Mosaic interface.

The mechanism can be considered similar to the one described by Rasmussen & Pirenne (1994) for Sybase, but in our case form definition files (FDFs) are never built, nor stored in the database. This design feature allows modification of the database structure without changing the archive software, keeping description files up-to-date, or maintaining additional software to be run on the user's computer.

To allow independence of the structure from the specific DBMS used, some conventions are enforced. First, no DBMS system table is directly used; a table called `$SYSTEMTAB` is used instead. This table contains the names of the various tables stored in the database, together with their fields, descriptions, and formats; information on joins and views are also stored. `$SYSTEMTAB` may be built from a DBMS system table by means of a specific utility.

Two additional tables must be available in the system. `$MOTHERTAB` contains the list of all tables and views which are explicitly available to the user when accessing the archive. `$PATHS` contains information on the location of both quick-look and original FITS data files.

Some additional naming conventions are enforced on fields. If a field in a table is called `$tablename`, it contains names of other tables stored in the database; this convention allows the creation of a hierarchy of database tables, if needed. The `$filename` field contains names of files stored in the archive, either on disk, on a juke box, or off-line. The `$hostname` field contains the name of a (remote) archive server. The `$path_id`, `$gif_path`, and `$fits_path` fields contain the location of quick-look and FITS data files in terms of directory paths; `$path_id` is given as a numerical value corresponding to a directory path, and is used to save space in the database.

4. Applications

The scheme developed allows the interface, and the overall system, to be completely independent from operating system, DBMS, archive structure, and database structure (except for the conventions described in the previous section).

These concepts have been tested on a prototype archive containing a few digitized objective prism images, related tables containing the position of detected spectra, and the extracted spectra themselves. Both original and quick look data are contained in the archive. Details on the implementation of the archive are given in Pasian & Smareglia (1994b). The related URL is SQL-ARCHIVE interface[1].

A distributed archive containing interstellar medium spectra is currently being designed (Porceddu et al. 1994), and an initial version should be available before the end of 1994. This archive will use the interface described here; its metadata will be contained in a database in Trieste, while the data will be stored in Cagliari, Pisa (Italy) and Torun (Poland). Finally, this architecture is planned to be used (Pasian 1994) as the interface to the technical archive of TNG, the Italian National 3.5 m telescope to be located at La Palma, in the Canary Islands.

Acknowledgments. M. Albrecht, A. Balestra, P. Marcucci, B. Pirenne, G. Russo, and C. Vuerli are gratefully acknowledged for many interesting discussions on the topics of archives and information retrieval.

References

Pasian, F. 1994, Archives at the TNG Telescope – Architecture Design Document, TNG Project Technical Report (draft 0.9)

Pasian, F., & Smareglia, R. 1994a, in New tools for network information retrieval, eds I. Porceddu, & S. Corda (Cagliari, Oss. Astr. Cagliari)

Pasian, F., & Smareglia, R. 1994b, Int. Jour. of Mod. Phys. C - Physics and Computers, in press

Porceddu, I., Corda, S., Pasian, F., & Smareglia, R., 1994, Proc. Conf. Boulder, Colorado, in press

Rasmussen, B. F. 1995, this volume, p. **??**

[1] http://atlantis.oat.ts.astro.it