Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

SAADA: An Automatic Archival System for Astronomical Data

Nguyen N.H., Michel L., Motch C.

Observatoire Astronomique de Strasbourg

Abstract. This papier presents an overview of SAADA, a tool designed to allow astronomers to easily create their own databases from archival files (images, spectra, tables, ...) or from imported data. It aims to make the process of database creation as automatic as possible. Its functionality will include java code generation, data loading, automatic web interfacing, and some interoperability features. Correlation links between records can easily be set up by astronomers in order to add scientific content to the database. Data can either be accessed with the automatic Web interface or by handling persistent objects. Through an API, SAADA will be able to interoperate with external databases (using of VO standards). It will also be able to achieve queries including constraints on correlation patterns.

1. Introduction

The increasing capabilities of both hardware and networking offer the possibility to astronomers to easily organize their own data in local databases. Nevertheless, setting up such databases remains difficult, especially for the complex and heterogeneous data used in astronomy. The presented development, SAADA, (Système Automatique d'Archivage de Données Astronomiques in French, Happiness in Arabic) aims at making the deployment of local databases easier.

SAADA is not a database system but a database generator. Databases created by SAADA will be hereafter refered as SAADA-DBs. All SAADA-DBs rest on the same branches of a common data model, but have their own object layers (API and Web interface) and their own relational bases.

The architecture of the SAADA-DBs tries to take advantage of both relational and object worlds. Object model is convenient to deal with heterogeneous data and to provide a simple API whereas relational database model is a mature solution to share large sets of information and to manage concurrency, transactions and roll-backs. A SAADA-DB is an object layer using a SQL RDBMS as repository. The goal of SAADA is to create a database system ready to use (a SAADA-DB) just by analysing input data and by applying some rules given by the data owner. SAADA relies on freeware, standard compliant, multiplatform and object oriented programming. SAADA is totally written in Java using number of public APIs (J2EE- Flanagan 1999). The purpose of this paper is more to explain the architecture of the SAADA-DBs than to describe the structure of SAADA itself (Figure 1).

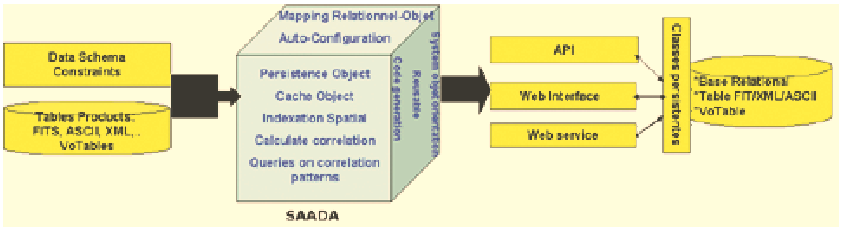

Figure 1. SAADA

2. Features of a SAADA-DB

A SAADA-DB can host tables, images, spectras and time series. Data are grouped in named collections (e.g. STARS, GALAXIES) set up by the owner. Once a new SAADA-DB is created and its data set loaded, it can be accessed through a web interface using servlets. The web interface provides browsing facilities and an editor for complex queries. It has functionnalities similar to those of the XCAT-DB interface (Michel 2004). A Java API allows database users to handle records in Java instances. The API is read-only by default but a specific mode can be used by the SAADA-DB's owner to set-up persistent links between data (e.g. cross-match). The API has not been designed to load data. This task is dedicated to the dataloader module by reading local products or by querying some external databases.

3. Architecture of a SAADA-DB

Below is a short description of the SAADA-DB modules. The organization of all SAADA-DB components is shown in Fig 2.

- **API specific:** Generated API including classes modelling the persistent data.

  Data loader: Module used to load data from the input files/streams.

- **Module servlet:** Automatically generated web interface.

- **Module web wervice:** This module handles database accesses by web services.

- **Object cache:** Achieves the conversion of relational data into persistent objects.

- **API generic:** Low level persistence functionalities.

- **High level query engine:** Query optimizer.

- **Module update:** This module is in charge of updating persistent objects. It can not create new objects but it can modify some attributes such as correlation links between instances.

- **Open modules:** Built out functionalities.

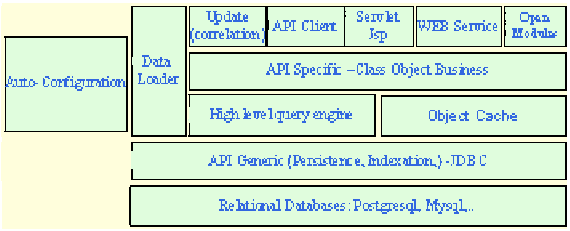

Figure 2. SAADA Architecture

4. Auto-configuration

One of the most useful property of SAADA , its auto-configurability, allows astronomers to create their own databases without writing any line of code in Java or SQL. With SAADA, database owners must just set a few rules at configuration time which specifies the mapping between input data and classes. Input data are identified by directory names, filename masks or some inner keywords. The collection in which data must be stored are also specified at configuration time. From this configuration file (XML) and from the input data checking, SAADA is able to build SQL tables and Java classes which together are going to form the new SAADA-DB.

5. Object-Relation Mapping

As all SAADA-DBs are built on the top of the same data model, the object mapping is quite simpler than for any other general purpose Object-Relationnal system. This simplicity added to considerations on low level functionality and on performances lead us to choose to develop our own object mapping layer. The mapping mechanism is classically (Rahayu 2000) based on the use of object identifiers (OIDs). From any OID SAADA is able to:

– determine the table where the object is.

  identify the object class.

– retrieve the instance content.

With a single OID, any data record can be retreived either in the relational world or in the object jungle (or vice-versa). OIDs are unique within a given SAADA-DB.

6. Performance fonctionnalites

6.1. Object Cache

The object cache is the hot spot of the system. It is in charge of transforming table records into Java instances and especially to minimize databse accesses. Its setup has a fundamental impact on global performance. Objects are handled by OIDs, their content is not read; but the first time an attribute is accessed,

the cache is invoked to build the full instance. Objects no longer referenced by the application are removed from the cache by the JVM garbage collector only when the memory heap is full.

Complex objects are built into the cache by applying a lazy-loading strategy (Kircher 2001).

6.2. Query Engine

Queries are processed by a separate module. The query optimization obviously has a significant impact on the global efficiency of a SAADA-DB. Users queries are translated into SQL queries including some built-in functions taking in account their complexity. Further local computation on SQL query results can be achieved before returning the final result. Queries only return sets of OIDs. Object contents can only be delivered by the cache. SAADA systematically implements into SAADA-DBs some specific features necessary to speed up queries. All data will be for instance referenced on a sky pixel map (e.g. Qbox, Page 2002) and specific indexes are setup for the processing of queries including constraints on correlated data patterns.

7. Development Status

A SAADA-DB is under test. It includes all of the basic functionnalities (cache, web interface). This prototype is built by a piece of software hosting the main modules of SAADA (auto-configuration, data loader). The first public distribution will be released in spring 2004. SAADA status can be seen at http://saada.u-strasbg.fr/

Acknowledgments. This project of thesis is funded by the Région Alsace (France) and by the Centre National d'Etudes Spatiales (CNES France).

References

Flanagan, D. 1999, Java Enterprise -In a nutshell, O'REILLY

Kircher, M. 2001, Lazy Acquisition, http://www.cs.wustl.edu/~mk1/LazyAcquisition.pdf

Michel, L., Motch, C., Page, C. G., Watson M. G. 2003, in ASP Conf. Ser., Vol. 295, ADASS XII, ed. H. E. Payne, R. I. Jedrzejewski, & R. N. Hook (San Francisco: ASP), 291

Michel, L., Motch, C., Pye, J., Watson, M. 2004, "XCAT-DB a Public Interface for the SSC XMM-Newton Catalogue' this volume, 570

Page, C. 2002, Indexing the Sky, http://wiki.astrogrid.org/bin/view/Astrogrid/SkyIndexing.

Rahayu, J.W. 2000, A method for transforming inheritance relationships in an object-oriented conceptual model to relational tables, ELSEVIER, Information and Software Technology 42(2000) 571-592.