# Practical Applications of a Relational Database of FITS Keywords

De Clarke, Steve Allen

*UCO/Lick Observatory, Kerr Hall, UCSC, 1156 High Street, Santa Cruz, CA 95064*

**Abstract.** In the course of designing a major mosaicked instrument (DEIMOS) we need tools to manage the large set of FITS keywords required for instrument control and data reduction/archiving. We have therefore constructed a relational database schema which describes FITS keywords (and have implemented it using Sybase). In this schema FITS keywords and structures are entities, having attributes such as format, units, datatype, minimum and maximum value, semantics, etc. The schema was quickly generalized to describe database tables as well as FITS headers and individual keywords. The schema was expanded to document and define software agents which handle the data. From the stored information, useful output products such as documentation, sample FITS headers, and diagrams of process and information flow can be generated by software. It also serves as an online documentary reference, via Web pages or local tools.

## 1. Introduction

Using a centralized database or information service to document FITS keywords is not a new idea. However, in previous incarnations of this concept that we have noted, the object has either been purely documentary or single-purpose (as in the SaveTheBits database schema, for example). Our goal was to capture enough information, and to generalize the information sufficiently, that multiple related applications could all use one authoritative, online reference database for information about FITS keywords. For example, live observing and engineering interfaces to an instrument could rely on the same database that was used to generate printed documents, drawings, and even sections of source code.

A more complete documention of our work, with demos and examples, can be found online at

`http://www.ucolick.org/~de/deimos/Memes.`

For information about the DEIMOS instrument, see

`http://www.ucolick.org/~loen/Deimos/deimos.html.`

## 2. Keywords and Memes

The first challenge was to determine the list of attributes necessary to document a FITS keyword. Some, such as name, FORTRAN datatype, semantics, and min/max values, were obvious. Others were more subtle, and the list of attributes was revised several times as we explored the syntax and semantics of existing FITS headers and our proposed DEIMOS instrument keywords.

We established immediately a need for attributes to model the following facts:

- FITS keywords have *provenance* or *context* (i.e. an institution, instrument, author, or standards document which defines their valid use). Knowing the context is essential, given that there is no constraint on duplication in FITS namespace.
- FITS keywords have relationships to other keywords in the same dataset (for example, the value of NAXIS controls the occurrence of NAXISn).
- FITS keywords occur in groups or "bundles"; a standard set of required keywords must appear in a valid table extension header, for example. A FITS header itself is a special case of such a bundle, as is a FITS table. We established ancillary tables to describe the grouping or bundling of keywords.

We found it necessary to establish special types of keywords called "tuple keywords," whose values are actually a list or other parsable structure of subvalues. We established a general-purpose way of documenting such tuple values, which handles any such keyword by defining subsemantics and subformats, separators, delimiters, etc. We accommodated the case where identical semantics are repeated N times, and the case where N submemes have distinct semantics.

Given that our instrument and telescope control systems use a keyword/value architecture, we needed to establish access control for keywords (not all of which ever appear in FITS headers); some are writable, some readable, some are both. The access control attributes of keywords are needed for the configuration of dynamically-generated graphical user interfaces. This was an extension to "strict" FITS keyword attributes.

We found it necessary to establish a special relationship between certain keywords and an "archetypical" keyword of the same semantics. For example, one instrument system may format a value as F8.4 where another will format it as F6.4 or as a string. It's evident that (where the semantics are identical) these are really "the same" keyword. We included attributes for defining and documenting this type of relationship, so that only one fully-defined archetypical keyword need be established, with minor variants which inherit most of their attributes from the original.

As we struggled with subsemantics and archetypes, it became clear that we were documenting something more abstract than "a FITS keyword." We borrowed the term "meme" to describe "a unit of meaning," a more general-purpose name for the entities we were manipulating.

This generalization led swiftly to the use of the "Memes" database to document itself, since the fields (columns) of a database table are memes with a large subset of the attributes of a FITS keyword (such as name, datatype, semantics, etc.) (The "archetypical" relationship described above was obviously ideal for expressing foreign keying between database tables.) Exploration of the relationship between database tables and Meme bundles led equally swiftly to automated translation between these formats.

In other words, we are able to extract (outload) a Sybase table as a FITS table extension, and to import (inload) a defined FITS table extension to a Sybase table. We are able to generate Sybase table definitions for any meme bundle, including standard FITS headers and FITS table extensions. There are many practical applications for these abilities, e.g. the automatic creation and population of database tables for user-defined event and telemetry logs, during instrument operation,

Although the page limit for this publication prevents us from documenting the Memes schema, we invite the FITS community to investigate the Memes table definition and related tables, using the demo reports available at

`http://www.ucolick.org/~de/deimos/Memes/demos.html`

If you choose "Documentation of Memes by Name" and enter the name Memes, the resulting report will be the current Memes table definition. We propose these attributes as the first draft of a standard set of attributes of FITS keywords, and invite comments from the FITS community.

## 3. Information Flow

The schema was expanded to include a set of ancillary tables describing protocols, formats, event timings, and agents (software, hardware, and human) so that we could document the flow of information through any large system such as the DEIMOS instrument control and data reduction package. The Agents schema permits us to document the source language, revision, authorship, and other attributes special to software agents. (This model of information flow corresponds well with the "gaming table" model proposed last year by Noordam and Deich.) As with Memes, a URI field permits the attachment of detailed documentation to any Agent.

A table of Paths documents the passage of Memes between Agents. The attributes of a Path are the sending and receiving agent, a controlling agent, the Meme ID, and a cluster of attributes describing the transaction: format, protocol, event timing, and elapsed time. Since human beings can be agents and Key Entry is an acceptable protocol, it is possible to document the Memes/Agents toolset itself using this schema.

We use a digraph generation package (see Acknowledgments) to generate graphical representations of information flow. It became obvious early in the design process that a hierarchy of agents and paths was required in order to generate both simplified and detailed drawings. Accordingly, the schema was adjusted to permit the definition of "superAgents" which consist of multiple Paths.

The Agents/Paths database can be used not only to generate functional diagrams, but to assist in diagnostic procedures. It can be used to trace the any FITS keyword value back to the originating or authoritative source; conversely, it can reveal which agents handle a keyword and thus may be affected by changes to the syntax or semantics of that keyword; it can reveal which keywords are handled by an agent, and which "downstream" agents may be affected if an agent is disabled or altered.

Currently we are addressing the distinction between syntactic and functional relationships among Memes. We have addressed, for example, the fact that the value of NAXIS controls the appearance of keywords NAXISn (a syntactic relationship). We have now to model those relationships in which a set of FITS keywords acts (in use, rather than in syntactic specification) as a state engine. This is particularly relevant for instrument control systems, in which (for example) a HATCHCLO keyword and a HATCHOPN keyword cannot both be true, but may both be false. The solution to this problem will almost certainly require one additional ancillary table in the schema.

## 4. Practical Applications

The Table of Output Products shows various practical applications, all of which use the same central database of Memes. Many of these applications are available as live demos at the URL cited above.

**Acknowledgments.** We would like to acknowledge the invaluable contributions of the free software community, with particular gratitude to Tom Poindexter (Talus Technologies) for the SybTcl package, Stephen North and Eleftherios Koutsoufios (Lucent) for their digraph toolkit, Per Cederqvist and friends for CVS, John Ousterhout (Sun) for Tcl/Tk, and Mark Diekhans and Karl Lehenbauer for TclX.

## References

Noordam, J. & Deich, W. 1978, ADASS V, 229

Table 1. Table of Output Products

| | Product | Practical Application |
|---|---|---|
| 1 | HTML documentation | We can generate (dynamic) documents describing individual keywords, database tables, FITS tables, and FITS headers. We plan to use such generated documents for design review requirements. These documents can be regenerated quickly whenever design decisions are changed. The same code generates documents of the database schema itself. |
| 2 | Sample FITS headers | We can generate sample (fake) FITS headers which can be used to test FITSIO code. |
| 3 | C source | We can generate C source for data structures (such as would be needed to store any given FITS header or table, and for FITSIO writer code (reader code is in progress). |
| 4 | Graphic FITS documentation | We can generate digraph-stype graphical representations of the structure of FITS headers, showing their use of individual keywords and pre-defined bundles. |
| 5 | Sybase DDL | We can generate Sybase table definitions (any other database engine could also be easily supported) for the dynamic creation of tables to store data from FITS headers. |
| 6 | FITS tables | We can generate conformant FITS table extensions containing data from Sybase tables; we can load data into Sybase tables from FITS table extensions. |
| 7 | Information Flow Graphs | We can generate flow charts documenting the movement of information between agents in a system. This generalized tool can document instrument control and data reduction; it also documents itself, or any other system which can be described by the passage of information elements between agents. |
| 8 | Online Access (code) | The online Meme database is being used as the live reference for DEIMOS interface prototypes. When the interface user wishes to monitor one or more keywords, the interface software generates a control panel with graphical meters and other widgets according to the attributes of the keywords being monitored. Keyword attributes need not be hardcoded into the interface, and the interface will always adapt itself to engineering changes if those changes are documented via the Memes database. |
| 9 | Online Access (user) | We expect to use the online database as a reference and documentary tool for users of the instrument, during actual observing. We expect that the dynamic generation of flow control graphs will assist in diagnosis of problems and improve user understanding of the instrument and its output. |