# GOM.FITS

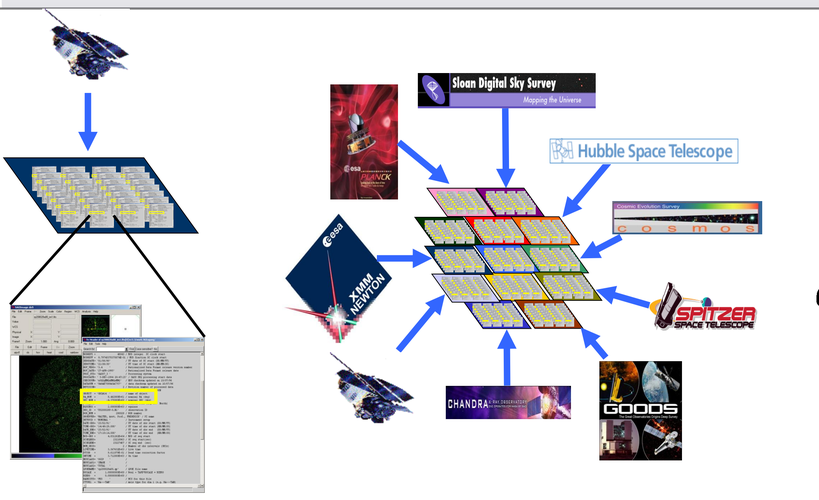
## Modelling and storing FITS metadata in a relational database.

Jai Won Kim (jkim@mpa-garching.mpg.de) and Gerard Lemson (lemson@mpa-garching.mpg.de)

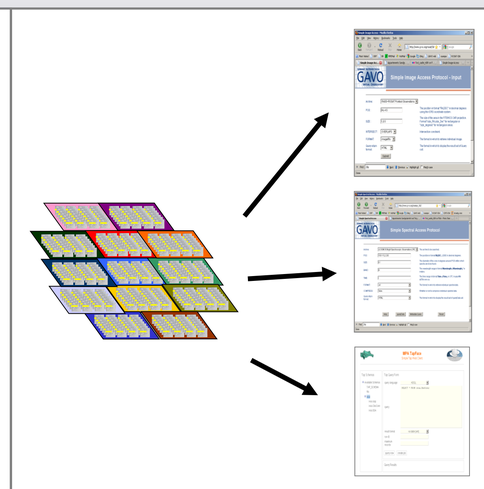
*Max-Planck-Institut für Astrophysik, Garching, Germany*

*Here we present GOM.FITS, a software solution for storing FITS metadata in a relational database according to a data model that treats every type of FITS file in a fully generic way. This solution has been used to publish various data sets through IVOA standards, but is generally useful in the management of large heterogeneous FITS catalogues, such as those derived from IVOA-like distributed queries. We motivate and describe the solution and show how to use it for discovering contents of such archives and how to use simple SQL queries to map the model contents to any standard specified by organizations such as the IVOA.*

## Motivation and Challenges




Most astronomical data such as images and spectra are stored in FITS[1] files. FITS provides data with their metadata. Some of the metadata is standardized, allowing the creation of tools for example to visualize images with their coordinates. But much of the metadata consists of custom keyword-value pairs. Different instruments produce FITS files with their own particular collection of headers. Such single-instrument archives may be uniform, but scientists generally need to access archives from many instruments. Collaborations often need to manage and share sets of heterogeneous FITS files, the structure of which may be unfamiliar.
Tools are needed to investigate such archives efficiently.



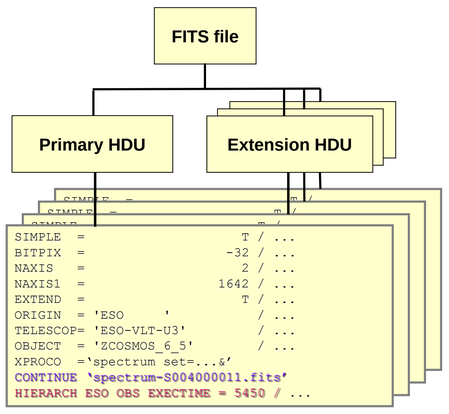

When publishing archives with FITS files to the Virtual Observatory, translations must be done from particular FITS metadata to standard models, some of which may overlap (e.g. SIAP[2], ObsTap[3]). These views of the FITS metadata must often be completed with service specific metadata not available in FITS headers. For example characterisation metadata derived from the data such as specified in the IVOA [4] Characterization Data Model [5].
There is need of a tool to support such complex mappings and allows for the computationof such missing metadata in a form that can be easily combined with the FITS headers.

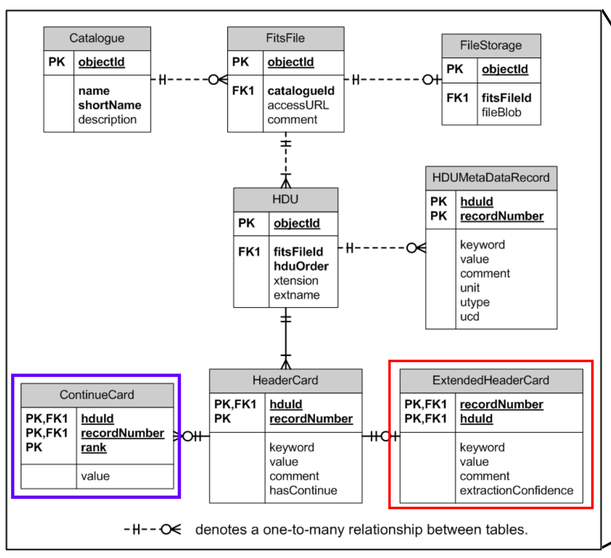
## Solution: GOM.FITS

**GOM.FITS** is our name for a solution to the above problems. We store FITS metadata in a relational database according to a generic data model. This relational model is a faithful representation of the FITS metadata structure. Rather than mapping a specific set of FITS keywords to columns in custom tables, header cards for all FITS files are mapped to rows in a single HeaderCard table. This allows one to store any type of FITS file in the same model, and allows one full freedom to generate any type of custom view by straightforward SQL queries.
We have implemented this solution in the *GOM.FITS Harvester*, a Java library that reads metadata from FITS files and stores them in the database.

### GOM.FITS Data Model

A faithful representation of FITS metadata structures as a relational model.

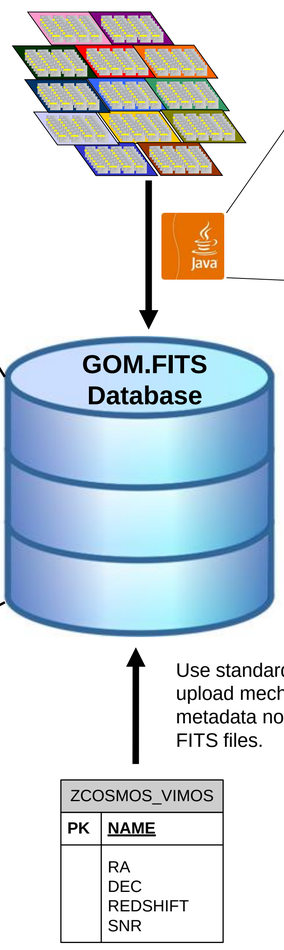



**Basic Model**
FITS file structures mapped almost 1-1 to database tables.

- FITS file → `FitsFile`
- Header Data Unit → `HDU`
- Header card → `HeaderCard`
  - CONTINUE → `ContinueCard`
  - HIERARCH → `ExtendedHeaderCard`

**Extras**:
HDUMetadataRecord: custom header cards derived from the FITS file using plug-ins
Catalogue : groups related FITS files
FileStorage : possibility to store FITS files as BLOBs in table. Or with access URL to remote location.




### GOM.FITS Harvester

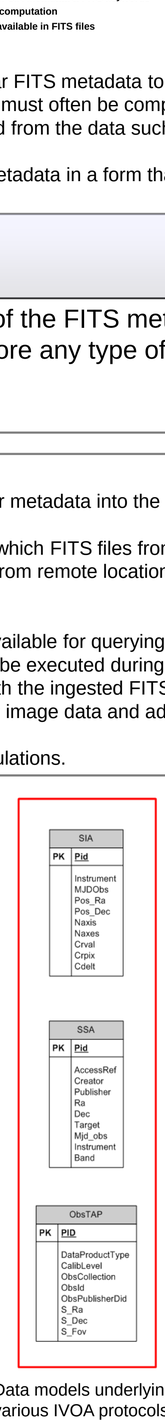
A Java library for reading directories of FITS files and storing their metadata into the GOM.FITS database.
Command line interface supporting regular expressions to select which FITS files from directory structure should be ingested. Also possible to load files from remote locations.
Possible to store FITS files as BLOBs in database.
Files are organised in catalogues that can be separately defined.
Run time logging metadata is stored in the same database and available for querying.
A plug-in mechanism is supported that allows custom methods to be executed during ingestion time to calculate extra metadata records to be stored with the ingested FITS file. For example characterisation metadata could be derived from image data and added to the corresponding HDU.
Standard plug-ins are available for example for WCS related calculations.

**SQL**

Standard SQL queries can be used against the GOM.FITS data model to browse the contents of FITS archives, "reporting" .

Also use standard SQL to map information from the GOM.FITS data model to an alternative representation, or model of choice.
In this mapping it is trivial also to add custom metadata not available in the FITS files, simply by joins to tables uploaded through standard database mechanisms.

Such models are best built as materialised views on the GOM.FITS model. For examples see next section.




Data models underlying various IVOA protocols

## Usage examples

### 1. Reporting query example

In many cases the metadata structure of a set of FITS files is unknown. **GOM.FITS**'s data model is perfectly suited for queries aimed at investigating the contents of the different files.

For example the query below finds all unique keyword-comment combinations and their multiplicity in the ROSAT image catalogue.

```
Select hc.keyword
     , hc.comment
     , count(*)
  From fits.Catalogue c
     , fits.FitsFile f
     , fits.Hdu h
     , fits.HeaderCard hc
 Where c.shortName = 'ROSAT'
   and c.objectId = f.catalogueId
   and f.objectId = h.fitsFileId
   and h.hduOrder = 0
   and h.objectId = hc.hduId
Group by hc.keyword
     , hc.comment
Order by hc.keyword
```

### 2. View generation using "pivoting SQL"

Often it is useful to produce representations of FITS metadata where keywords are used to define columns. Compared to the GOM.FITS model, the tables are as it were *pivoted*. SQL is rich enough to support such operations, where usually one will want to store the result as a materialized view.

**TABLE_A**

| ID | KEYWORD | VALUE |
|---|---|---|
| 1 | INSTRUME | PSPCC |
| 1 | RA_NOM | 3.32E+02 |
| 1 | DEC_NOM | 4.65E+01 |
| 2 | INSTRUME | PSPCB |
| 2 | RA_NOM | 7.60E-01 |
| 2 | DEC_NOM | 6.26E+01 |

Pivot Table SQL

**TABLE_B**

| ID | INSTRUME | RA_NOM | DEC_NOM |
|---|---|---|---|
| 1 | PSPCC | 3.32E+02 | 4.65E+01 |
| 2 | PSPCB | 7.60E-01 | 6.26E+01 |
| 3 | HRI | 3.80E+01 | 8.93E+01 |
| 4 | HRI | 3.80E+01 | 8.93E+01 |
| 5 | PSPCB | 3.44E+00 | 7.29E+01 |

```
Insert Into TABLE_B
Select a.ID as ID
     , MAX(Case When a.KEYWORD='INSTRUME'
          Then a.VALUE Else null End) as INSTRUME
     , MAX(Case When a.KEYWORD='RA_NOM'
          Then a.VALUE Else null End) as RA_NOM
     , MAX(Case When a.KEYWORD='DEC_NOM'
          Then a.VALUE Else null End) as DEC_NOM
  From TABLE_A a
 Group by a.ID
```

### 3. Example SQL mapping to IVOA metadata

Identifying ObsTAP/ObsCore metadata[3] from ROSAT All Sky Survey.

```
Insert Into ivoa.ObsCore(...)
Select r.catalogueName
 , 'Image.2D' as dataproduct_type
 , r.access_url, r.target_name
 , r.s_ra  ,r. s_dec
 , 'BOX '+r.radecsys +' '+cast(r.s_ra as decimal(8,3))+' '
   +cast(r.s_dec as decimal(8,3))+' 6.0 6.0'  as region
 , (r.scseqend - r.scseqbeg) as t_exptime
 , ...
From (
  Select f.objectid as fitsFileId, h.objectid as hduId
     , MAX(c.name) as catalogueName
     , MAX(f.accessURL) as access_url
     , MAX(Case When hc.keyword='OBJECT'
            Then hc.value Else NULL End) as target_name
     , MAX(Case When hc.keyword='OBS_MODE'
            Then hc.value Else NULL End) as obs_mode
     , MAX(Case When hc.keyword='RADECSYS'
            Then hc.value Else NULL End) as radecsys
     , MAX(Case When hc.keyword='RA_NOM'
            Then cast(hc.value as float) Else NULL End) as s_ra
     , MAX(Case When hc.keyword='DEC_NOM'
            Then cast(hc.value as float) Else NULL End) as s_dec
     , MAX(Case When hc.keyword='SCSEQBEG'
            Then cast(hc.value as int)Else NULL End) as scseqbeg
     , MAX(Case When hc.keyword='SCSEQEND'
            Then cast(hc.value as int) Else NULL End) as scseqend
  From fits.catalogue c, fits.fitsFile f
  ,    fits.hdu h, fits.headercard hc
  Where c.shortName = 'ROSAT'     and c.objectId = f.catalogueId
    and f.objectId = h.fitsFileId and h.hduOrder = 0
    and h.objectId = hc.hduId
  Group by h.objectId, f.objectId, c.objectId   ) r
Where r.obs_mode = 'SURVEY'
```

References
[1] W. D. Pence, *et al*, A&A 524, A42 (2010)
[2] http://www.ivoa.net/Documents/SIA/20091116/REC-SIA-1.0.pdf
[3] http://www.ivoa.net/Documents/ObsCore/20111008/PR-ObsCore-v1.0-20111008.pdf
[4] http://www.ivoa.net
[5] http://www.ivoa.net/Documents/latest/CharacterisationDM.html

For further details
http://gavo.mpe.mpg.de/GOM.FITS