The SIMPLE Archive

David Rodriguez[1], Kelle Cruz[2], Will Cooper[3], Niall Whiteford[4], Clemence Fontanive[5], Ella Hort[6], Sherelyn Alejandro[2], Robert Blackwell[7], Daniel Terach[8]

[1]*Space Telescope Science Institute, Baltimore, MD, USA;* drodriguez@stsci.edu

[2]*CUNY Hunter College, New York, NY, USA*

[3]*University of Hertfordshire, Hertfordshire, United Kingdom*

[4]*University of Edinburgh, Edinburgh, Scotland, United Kingdom*

[5]*CSH, University of Bern, Bern, Switzerland*

[6]*Pomona College, Claremont, CA, USA*

[7]*Flatiron Institute, New York, NY, USA*

[8]*Pace University, New York, NY, USA*

Abstract.

We present the SIMPLE Archive alongside its database management tool, AstrodbKit2. SIMPLE is an archive of low mass stars, brown dwarfs, and exoplanets driven by community curation and review using GitHub. SIMPLE relies on AstrodbKit2 to convert back and forth from a document-store model of the database, to a more standard relational database that can be used with established packages like SQLAlchemy. In this poster, we present the architecture of the SIMPLE database and how using AstrodbKit2 facilitates a git workflow for reviewing and approving database modifications.

SIMPLE is available at https://github.com/SIMPLE-AstroDB/SIMPLE-db
AstrodbKit2 is available at https://github.com/dr-rodriguez/AstrodbKit2

1. SIMPLE

SIMPLE is an archive of low mass stars, brown dwarfs, and exoplanets. The vision behind SIMPLE is that of a community-driven archive were individual contributors help grow the knowledge base of these objects. SIMPLE contains a variety of data and supporting metadata for these low mass objects, but serves as an example that can be adapted to other fields of study, such as extragalactic or supernova archives. SIMPLE strives to represent each individual source as a consistent object that can be understood and referenced by others during the community review process. To that end, supporting software (AstrodbKit2) had to be created to facility this workflow.

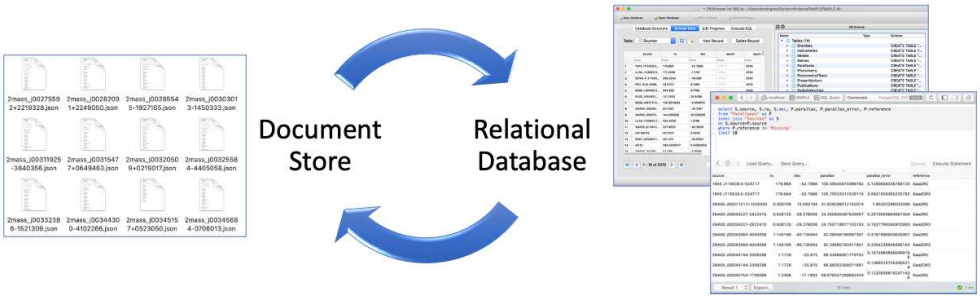

Figure 1. AstrodbKit2 can convert a database from a document store representation (eg, a list of JSON files) to a relational database (eg, SQLite, Postgres, etc).

2. AstrodbKit2

AstrodbKit2 is a Python package that uses SQLAlchemy to create and connect to a variety of relational databases (eg, SQLite, Postgres, MSSQL, etc). Tables in AstrodbKit2 are organized into two types: *Object* tables, which have one-to-many relationships to a single primary object table (ie, Sources in the SIMPLE database); and *Reference* tables, which have many-to-many relationships against the object tables and are used to store lookup information like publications, telescopes, or instruments. The SIMPLE Archive gathers measurements for low mass objects into a variety of tables all associated to the primary Sources table by their source name. Object tables in SIMPLE include Photometry, Proper Motions, Radial Velocities, and Spectra; Reference tables include Publications, Telescopes, and Instruments.

Using AstrodbKit2, we can convert the SIMPLE database from a document store mode, where individual Sources are stored as separate JSON files (which can in turn be loaded into NoSQL databases like MongoDB), to a relational database such as SQLite or Postgres that can be accessed with standard tools (see Figure 1). JSON files, such as the example in Figure 2 serve as the definitive copy of the database for purposes of version control.

3. Collaborative Workflow

By exporting a database to a JSON document store, we can use git and GitHub to handle version control for our database as well as curate commits via pull requests. An individual user may contain their own copy of SIMPLE, or any other database supported by AstrodbKit2. They may make changes in their local branch and push to their copy on GitHub. By issuing a pull request they request their changes be adopted into the main branch of the database. Because the database is stored as individual JSON documents, reviewers can see exactly which objects have been updated and can comment on the changes if needed. By using a simple to read format, we benefit from a clear picture of the changes in a pull request.

As part of the pull request process, automatic tests are run to verify the integrity of the database. This ensures no changes took place that break the functionality of the

database. Finally, when the pull request is accepted, additional automated tasks can be performed to regenerate the database and push it to external users of the database, such as a graphical user interface.

```
{
"Sources": [
    {
        "source": "2MASS J11040127+1959217",
        "ra": 166.005291,
        "dec": 19.989361,
        "epoch": null,
        "equinox": null,
        "shortname": "1104+1959",
        "reference": "Cruz03",
        "comments": null
    }
],
"Names": [
    {
        "other_name": "2MASS J11040127+1959217"
    },
    {
        "other_name": "Gaia DR2 3985153334796556928"
    }
],
"Photometry": [
    {
        "band": "2MASS.J",
        "ucd": "em.IR.J",
        "magnitude": 14.38,
        "magnitude_error": 0.026,
        "telescope": "2MASS",
        "instrument": "2MASS",
        "epoch": null,
        "comments": null,
        "reference": "Cutr03"
    },
    {
        "band": "2MASS.Ks",
        "ucd": "em.IR.K",
        "magnitude": 12.95,
        "magnitude_error": 0.029,
        "telescope": "2MASS",
        "instrument": "2MASS",
        "epoch": null,
        "comments": null,
        "reference": "Cutr03"
    }
]
}
```


Figure 2. Trimmed JSON document for a source in the SIMPLE database