The SIMPLE Archive

David R. Rodriguez (STScI), Kelle Cruz (CUNY Hunter College), Will Cooper (University of Hertfordshire), Niall Whiteford (University of Edinburgh), Clemence Fontanive (CSH, University of Bern), Ella Hort (Pomona College), Sherelyn Alejandro (Hunter College), Robert Blackwell (Flatiron Institute), Daniel Terach (Pace University)

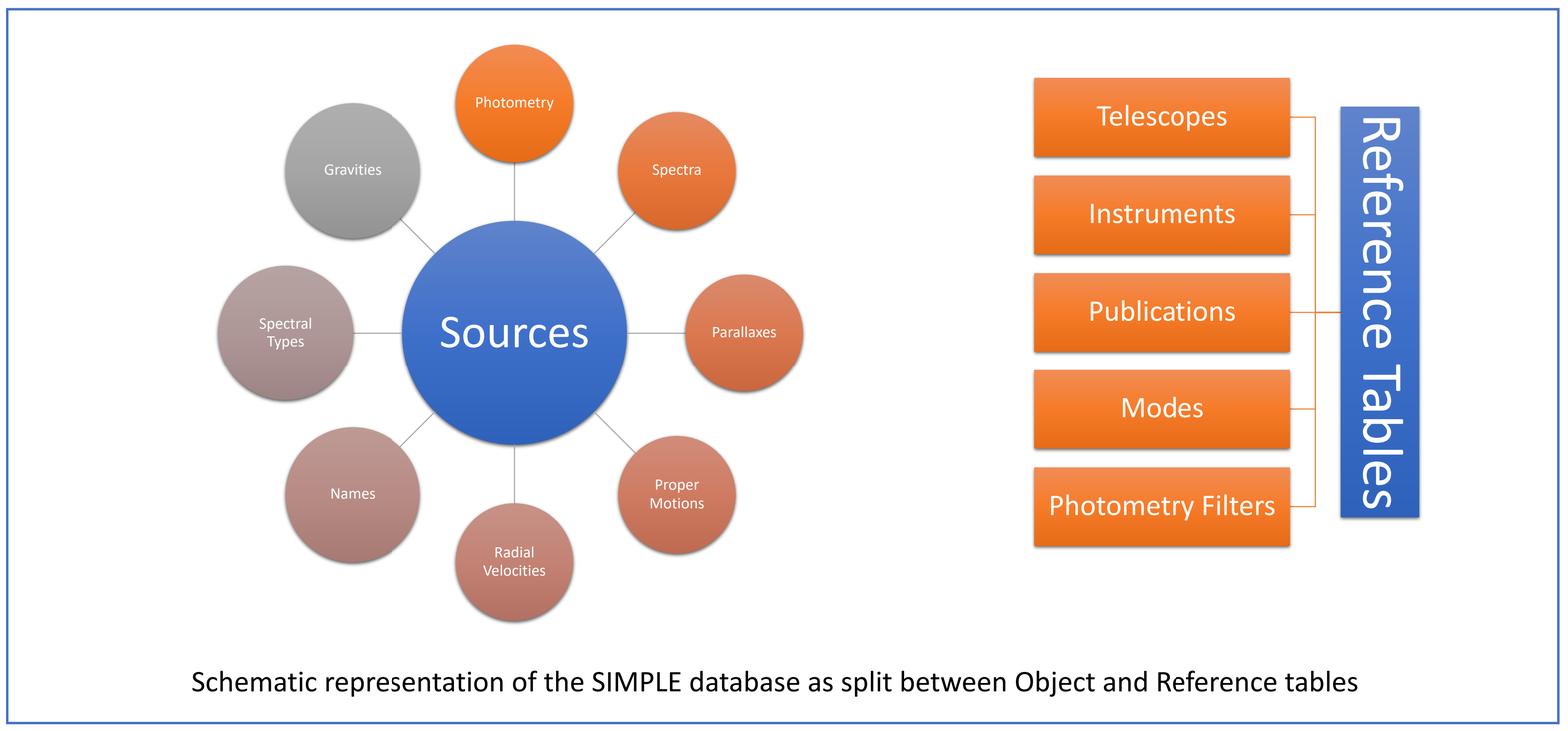
We present the SIMPLE Archive alongside its database management tool, AstrodbKit2. SIMPLE is an archive of low mass stars, brown dwarfs, and exoplanets driven by community curation and review using GitHub. SIMPLE relies on AstrodbKit2 to convert back and forth from a document-store model of the database, 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

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, as can be seen below.

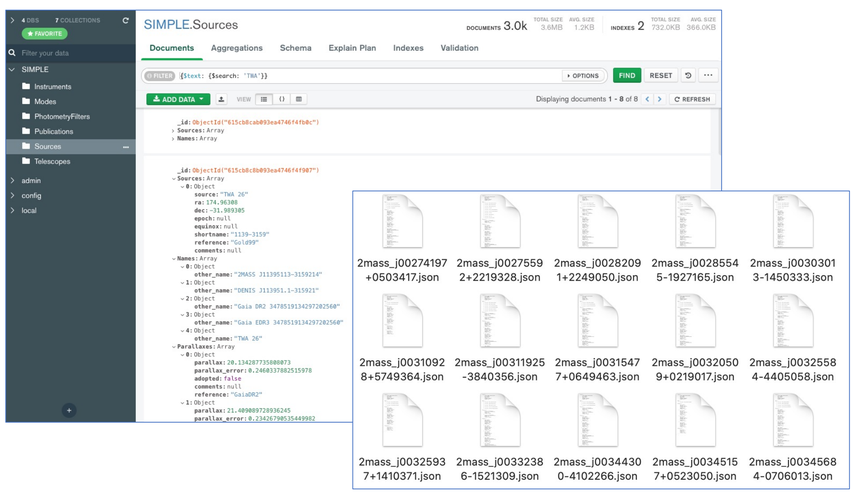
```python
# ----------------------------------------------------------------------
# Main tables
class Sources(Base):
    """ORM for the sources table. This stores the main identifiers for our objects along with ra and dec"""
    __tablename__ = 'Sources'
    source = Column(String(100), primary_key=True, nullable=False)
    ra = Column(Float)
    dec = Column(Float)
    epoch = Column(Float)  # decimal year
    equinox = Column(String(10))  # eg, J2000
    shortname = Column(String(30))  # not needed?
    reference = Column(String(30), ForeignKey('Publications.name', onupdate='cascade'), nullable=False)
    comments = Column(String(1000))


class Names(Base):
    __tablename__ = 'Names'
    source = Column(String(100), ForeignKey('Sources.source', ondelete='cascade', onupdate='cascade'),
                    nullable=False, primary_key=True)
    other_name = Column(String(100), primary_key=True, nullable=False)


class Photometry(Base):
    __tablename__ = 'Photometry'
    source = Column(String(100), ForeignKey('Sources.source', ondelete='cascade', onupdate='cascade'),
                    nullable=False, primary_key=True)
    band = Column(String(30), primary_key=True)
    ucd = Column(String(100))
    magnitude = Column(Float, nullable=False)
    magnitude_error = Column(Float)
    telescope = Column(String(30))
    instrument = Column(String(30))
```

Snippet of SIMPLE schema as defined with SQLAlchemy



Schematic representation of the SIMPLE database as split between Object and Reference tables

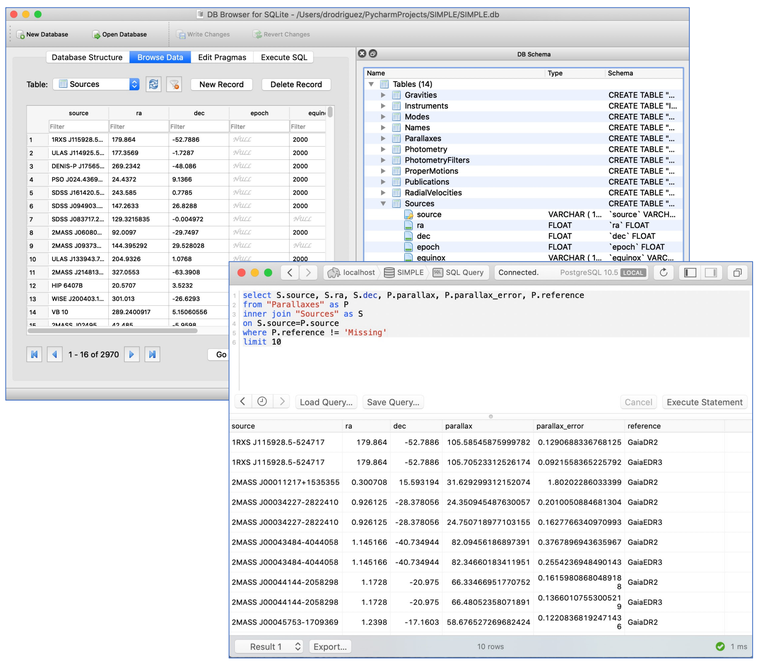
Using AstrodbKit2, we can convert the SIMPLE database from a document store mode, where individual Sources are stored as JSON files (which can in turn be loaded into NoSQL databases like MongoDB), to a relational database such as SQLite or Postgres that can be accessed with standard tools. JSON files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



Examples of handling SIMPLE as a document store

Document Store

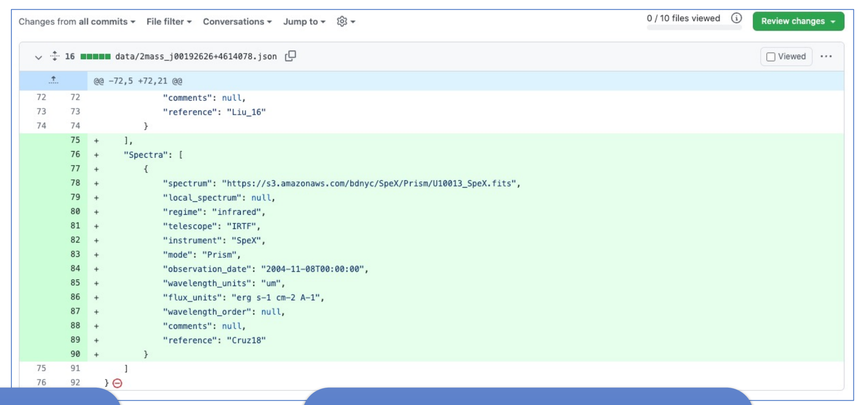
Relational Database

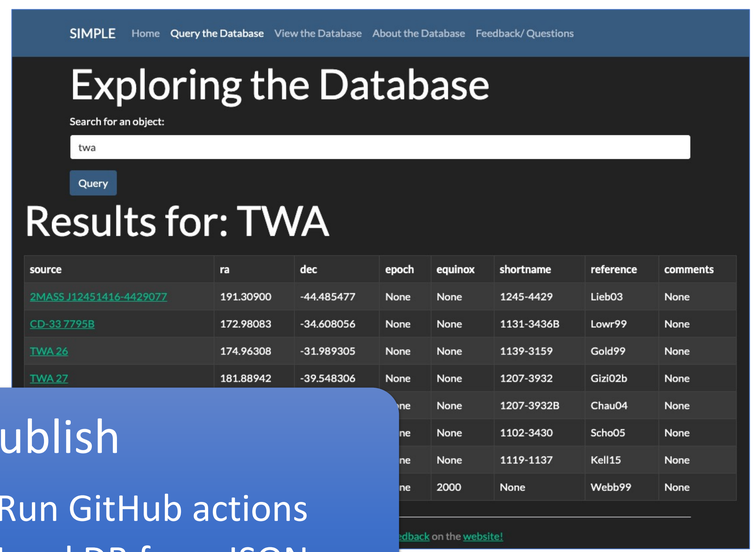


Examples of handling SIMPLE as a relational database

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. The chart below illustrates the workflow from inserting new data to a local database instance via Python, review of database modifications in GitHub, and then pushing changes to external uses of the database, such as a GUI.

```python
# Add source
sources_data = [{'ra': 209.301675, 'dec': 14.477722,
                 'source': '2MASS J13571237+1428398',
                 'reference': 'Schm10',
                 'shortname': '1357+1428'}]
db.Sources.insert().execute(sources_data)

# Additional names
names_data = [{'source': '2MASS J13571237+1428398',
               'other_name': 'SDSS J135712.40+142839.8'},
              {'source': '2MASS J13571237+1428398',
               'other_name': '2MASS J13571237+1428398'},
              ]
db.Names.insert().execute(names_data)

# Add Photometry
phot_data = [{'source': '2MASS J13571237+1428398',
              'band': 'WISE_W1',
              'magnitude': 13.348,
              'magnitude_error': 0.025,
              'telescope': 'WISE',
              'reference': 'Cutr12'
              },
             {...},
             {...},
             {...}]
db.Photometry.insert().execute(phot_data)

# Checking object
_ = db.inventory('2MASS J13571237+1428398', pret

# Save single object
db.save_json('2MASS J13571237+1428398', 'data')

# Save entire database to directory 'data'
db.save_database('data')
```





Develop

- Work on local instance
- Export to JSON
- Commit to branch

Review

- Issue pull request
- Run validation tests
- Merge changes

Publish

- Run GitHub actions
- Load DB from JSON
- Deploy to website