Introducing Records: SQL for Humans™

2016

2 min read • 372 words

Records is a very simple, but powerful, library for making raw SQL queries to Postgres databases.

This common task can be surprisingly difficult with the standard tools available. This library strives to make this workflow as simple aspossible, while providing an elegant interface to work with your queryresults—continuing the "for Humans" design philosophy explored in Ahead of My Time, I Think.

We know how to write SQL, so let's send some to our database:

```
import records

db = records.Database('postgres://...')
rows = db.query('select * from active_users')
```

The Basics

Rows are represented as standard Python dictionaries: `{'column-name': 'value'}`.

Grab one row at a time:

```
>>> rows.next()
{'username': 'hansolo', 'name': 'Henry Ford', 'active': True, 'timezone': datetime.date
```

Or iterate over them:

```
for row in rows:
    spam_user(name=row['name'], email=row['user_email'])
```

Or store them all for later reference:

```
>>> rows.all()
[{'username': ...}, {'username': ...}, {'username': ...}, ...]
```

Features

- HSTORE support, if available.
- Iterated rows are cached for future reference.
- `$DATABASE_URL` environment variable support.
- Convenience `Database.get_table_names` method.
- Queries can be passed as strings or filenames, parameters supported.
- Safe parameterization: `Database.query('life=%s', params=('42',))`
- Query results are iterators of standard Python dictionaries: `{'column-name': 'value'}`

Records is proudly powered by Psycopg2 and Tablib.

Data Export Functionality

Records also features full Tablib integration (my first popular project!), and allows you to export your results to CSV, XLS, JSON, or YAML with a single line of code.

Excellent for sharing data with friends, or generating reports.

```
>>> print rows.dataset
username|active|name      |user_email       |timezone
--------|------|----------|-----------------|---------------------------
hansolo |True  |Henry Ford|hansolo@gmail.com|2016-02-06 22:28:23.894202
...
```

Export your query results to CSV:

```
>>> print rows.dataset.csv
username,active,name,user_email,timezone
hansolo,True,Henry Ford,hansolo@gmail.com,2016-02-06 22:28:23.894202
...
```

YAML:

```
>>> print rows.dataset.yaml
- {active: true, name: Henry Ford, timezone: '2016-02-06 22:28:23.894202', user_email:
...
```

JSON:

```
>>> print rows.dataset.json
[{"username": "hansolo", "active": true, "name": "Henry Ford", "user_email": "hansolo@g
```

Excel:

```
with open('report.xls', 'wb') as f:
    f.write(rows.dataset.xls)
```

You get the point. Of course, all other features of Tablib are also available, so you can sort results, add/remove columns/rows, remove duplicates, transpose the table, add separators, slice data by column,and more.

See the Tablib Documentationfor more details.

Installation

Of course, the recommended installation method is pip:

```
$ pip install records
```

Hyperlinks

- Records on GitHub
- Records on PyPi
- Tablib Documentation
- Psycopg2 Documentation