# python-pgextras Documentation

***Release 0.2.1***

**Scott Woodall**

December 01, 2018

Contents

Contents:

CHAPTER 1

# python-pgextras

Unofficial Python port of Heroku's pgextras that provides various statistics for a Postgres instance. The biggest difference of this module is that your Postgres database can be anywhere, not just hosted with Heroku.

- BSD license
- Tested against 2.7 and 3.3
- Documentation
- Github

## Features

Note: pgextras does not format the output as seen in all the examples below. Instead it returns an iterable that contains namedtuples. For example, the "Total Indexes Size" method returns the following:

```
[
    Record(table='pgbench_accounts', index_size='2208 kB'),
    Record(table='pgbench_tellers', index_size='16 kB'),
    Record(table='pgbench_branches', index_size='16 kB'),
    Record(table='pgbench_history', index_size='0 bytes')
]
```

It's up to you on how to format and present the data. The results below are formatted to make it easier to understand the data available to you.

## Dev Install

**This is specific to the ::** update-sql-statements branch

Clone the repository and change the branch to *update-sql-statements* cd in to the directory and run

```
python setup.py install
```

This should install the pgextras exectutable

Now, you can use any postgres database to run pgextras

## Cache Hit

Calculates your cache hit rate (effective databases are at 99% and up):

```
| name           |    ratio |
|----------------+----------|
| index hit rate | 0.999888 |
| table hit rate | 0.999696 |
```

## Index Usage

Calculates your index hit rate (effective databases are at 99% and up):

```
| relname          |   percent_of_times_index_used |   rows_in_table |
|------------------+-------------------------------+-----------------|
| pgbench_history  |                               |          149985 |
| pgbench_accounts |                            99 |          100000 |
| pgbench_tellers  |                            96 |              10 |
| pgbench_branches |                            93 |               1 |
```

## Calls

Show 10 most frequently called queries:

```
| qry                                    | exec_time      | prop_exec_time | ncalls   | sync_io_time
|----------------------------------------+----------------+----------------+----------+-------------
| BEGIN;                                 | 0:00:00.140968 | 0.0%           | 414000   | 0:00:00
| INSERT INTO pgbench_history (tid, bid.. | 0:00:03.788899 | 0.0%           | 414000   | 0:00:00
| SELECT abalance FROM pgbench_accounts.. | 0:00:04.471754 | 0.0%           | 414000   | 0:00:00
| UPDATE pgbench_accounts SET abalance .. | 0:00:21.798029 | 0.2%           | 414000   | 0:00:00
| END;                                   | 0:00:00.126220 | 0.0%           | 414000   | 0:00:00
| UPDATE pgbench_branches SET bbalance .. | 0:30:00.544749 | 15.9%          | 414000   | 0:00:00
| UPDATE pgbench_tellers SET tbalance =.. | 2:38:45.396566 | 83.9%          | 414000   | 0:00:00
| BEGIN                                  | 0:00:00.002141 | 0.0%           | 212      | 0:00:00
| SELECT pid, application_name AS sourc.. | 0:00:00.039576 | 0.0%           | 77       | 0:00:00
| SELECT exists( SELECT ? FROM pg_exten.. | 0:00:00.001085 | 0.0%           | 43       | 0:00:00
```

## Blocking

Display queries holding locks other queries are waiting to be released:

```
|   blocked_pid | blocking_statement                                                                  | blocking_du
|---------------+-------------------------------------------------------------------------------------+------------
|          1688 | UPDATE pgbench_tellers SET tbalance = tbalance + -2309 WHERE tid = 9; | 0:00:00.018
```

## Outliers

Show 10 queries that have longest execution time in aggregate:

```
| qry                                    | exec_time      | prop_exec_time | ncalls   | sync_io_time
|----------------------------------------+----------------+----------------+----------+-------------
| UPDATE pgbench_tellers SET tbalance =.. | 2:59:30.137916 | 83.9%          | 467897   | 0:00:00
```

```
| UPDATE pgbench_branches SET bbalance .. | 0:33:53.945889 | 15.8%          | 467856   | 0:00:00
| UPDATE pgbench_accounts SET abalance .. | 0:00:25.384166 | 0.2%           | 467897   | 0:00:00
| SELECT abalance FROM pgbench_accounts.. | 0:00:05.086917 | 0.0%           | 467897   | 0:00:00
| INSERT INTO pgbench_history (tid, bid.. | 0:00:04.356031 | 0.0%           | 467848   | 0:00:00
| vacuum pgbench_branches                 | 0:00:00.336647 | 0.0%           | 17       | 0:00:00
| select count(*) from pgbench_accounts ; | 0:00:00.294740 | 0.0%           | 1        | 0:00:00
| BEGIN;                                  | 0:00:00.160855 | 0.0%           | 467897   | 0:00:00
| END;                                    | 0:00:00.142983 | 0.0%           | 467848   | 0:00:00
| SELECT relname, CASE idx_scan WHEN ? .. | 0:00:00.110683 | 0.0%           | 6        | 0:00:00
```

## Vacuum Stats

Show dead rows and whether an automatic vacuum is expected to be triggered:

```
| schema   | table            | last_vacuum      | last_autovacuum   | rowcount   | dead_rowcount
|----------+------------------+------------------+-------------------+------------+-----------------
| public   | pgbench_tellers  | 2014-04-24 20:02 | 2014-04-24 20:03  | 10         | 0
| public   | pgbench_branches | 2014-04-24 20:02 | 2014-04-24 20:03  | 1          | 0
| public   | pgbench_history  | 2014-04-23 20:45 |                   | 15000      | 0
| public   | pgbench_accounts | 2014-04-23 20:45 |                   | 100000     | 17581          |
```

## Bloat

Table and index bloat in your database ordered by most wasteful:

```
| type   | schemaname   | object_name                                    | bloat | waste        |
|--------+--------------+------------------------------------------------+-------+--------------|
| table  | public       | pgbench_accounts                               | 1.3   | 3768 kB      |
| table  | public       | pgbench_tellers                                | 19    | 144 kB       |
| table  | public       | pgbench_branches                               | 8     | 56 kB        |
```

## Long Running Queries

Show all queries running longer than five minutes by descending duration:

```
| pid   |    duration     |                                        query
|-------+-----------------+-----------------------------------------------------------------------------
| 19578 | 02:29:11.200129 | EXPLAIN SELECT  "students".* FROM "students" WHERE "students"."id" = 1450
| 19465 | 02:26:05.542653 | EXPLAIN SELECT  "students".* FROM "students" WHERE "students"."id" = 1889
| 19632 | 02:24:46.962818 | EXPLAIN SELECT  "students".* FROM "students" WHERE "students"."id" = 1581
```

## Sequence Scans

Show the count of sequential scans by table descending by order:

```
| name             |   count |
|------------------+---------|
| pgbench_branches |   57086 |
| pgbench_tellers  |   15595 |
| pgbench_accounts |       2 |
| pgbench_history  |       0 |
```

## Unused Indexes

Show unused and almost unused indexes, ordered by their size relative to the number of index scans. Exclude indexes of very small tables (less than 5 pages), where the planner will almost invariably select a sequential scan, but may not in the future as the table grows:

```
| table               |                        index                 | index_size | index_scans |
|---------------------+----------------------------------------------+------------+-------------|
| public.grade_levels | index_placement_attempts_on_grade_level_id | 97 MB      |           0 |
| public.observations | observations_attrs_grade_resources         | 33 MB      |           0 |
| public.messages     | user_resource_id_idx                       | 12 MB      |           0 |
```

## Total Table Size

Show the size of the tables (including indexes), descending by size:

```
| name             | size    |
|------------------+---------|
| pgbench_accounts | 18 MB   |
| pgbench_history  | 2904 kB |
| pgbench_tellers  | 272 kB  |
| pgbench_branches | 256 kB  |
```

## Total Indexes Size

Show the total size of all the indexes on each table, descending by size:

```
| table            | index_size   |
|------------------+--------------|
| pgbench_accounts | 2208 kB      |
| pgbench_tellers  | 16 kB        |
| pgbench_branches | 16 kB        |
| pgbench_history  | 0 bytes      |
```

## Table Size

Show the size of the tables (excluding indexes), descending by size:

```
| name             | size    |
|------------------+---------|
| pgbench_accounts | 16 MB   |
| pgbench_history  | 2904 kB |
| pgbench_tellers  | 256 kB  |
| pgbench_branches | 240 kB  |
```

## Index Size

Show the size of indexes, descending by size:

```
| name                  | size    |
|-----------------------+---------|
| pgbench_accounts_pkey | 2208 kB |
```

```
| pgbench_tellers_pkey  | 16 kB   |
| pgbench_branches_pkey | 16 kB   |
```

## Total Index Size

Show the total size of all indexes:

```
| size    |
|---------|
| 2240 kB |
```

## Locks

Display queries with active locks:

```
| procpid | relname | transactionid | granted |     query_snippet     |       age
|---------+---------+---------------+---------+-----------------------+-----------------
| 31776   |         |               | t       | <IDLE> in transaction | 00:19:29.837898
| 31776   |         |          1294 | t       | <IDLE> in transaction | 00:19:29.837898
| 31912   |         |               | t       | select * from hello;  | 00:19:17.94259
| 3443    |         |               | t       |                      +| 00:00:00
|         |         |               |         | select               +|
|         |         |               |         | pg_stat_activi        |
```

## Table Indexes Size

Show the total size of all the indexes on each table, descending by size:

```
| table            | index_size   |
|------------------+--------------|
| pgbench_accounts | 2208 kB      |
| pgbench_tellers  | 16 kB        |
| pgbench_branches | 16 kB        |
| pgbench_history  | 0 bytes      |
```

## PS

View active queries with execution time:

```
|   pid | source   | running_for             |   waiting | query
|-------+----------+-------------------------+-----------+-----------------------------------------
| 28023 | pgbench  | 0:00:00.107013          |         0 | UPDATE pgbench_accounts SET abalance = aba
| 28018 | pgbench  | 0:00:00.017257          |         0 | END;
| 28015 | pgbench  | 0:00:00.001055          |         1 | UPDATE pgbench_branches SET bbalance = bba
```

## Version

Get the Postgres server version:

```
| version
|-----------------------------------------------------------------------------------------------------------
| PostgreSQL 9.3.3 on x86_64-apple-darwin13.0.0, compiled by Apple LLVM version 5.0 (clang-500.2.79)
```

CHAPTER 2

# Installation

At the command line:

```
$ pip install pgextras
```

CHAPTER 3

# Usage

Your `dsn` value will be specific to the Postgres database you want to connect to. See the Postgres documentation for more information on configuring connection strings:

```
>>> from pgextras import PgExtras
>>> with PgExtras(dsn='dbname=testing') as pg:
...     results = pg.bloat()
...     for row in results:
...         print(row)
...
Record(type='index', schemaname='public', object_name='addresses_to_geocode::addresses_to_geocode_ado
Record(type='table', schemaname='public', object_name='addresses_to_geocode', bloat=Decimal('1.2'), w
Record(type='table', schemaname='pg_catalog', object_name='pg_attribute', bloat=Decimal('2.5'), waste
```

Or from the CLI:

```
$ pgextras -dsn "dbname=testing" -methods bloat version
```

## Class Methods

### .bloat()

```
    def bloat(self):
        """
        Table and index bloat in your database ordered by most wasteful.

        Record(
            type='index',
            schemaname='public',
            object_name='pgbench_accounts::pgbench_accounts_pkey',
            bloat=Decimal('0.2'),
            waste='0 bytes'
        )

        :returns: list of Records
        """

        return self.execute(sql.BLOAT)
```

## .blocking()

```
def blocking(self):
    """
    Display queries holding locks other queries are waiting to be
    released.

    Record(
        pid=40821,
        source='',
        running_for=datetime.timedelta(0, 0, 2857),
        waiting=False,
        query='SELECT pg_sleep(10);'
    )

    :returns: list of Records
    """

    return self.execute(
        sql.BLOCKING.format(
            query_column=self.query_column,
            pid_column=self.pid_column
        )
    )
```

## .cache_hit()

```
def cache_hit(self):
    """
    Calculates your cache hit rate (effective databases are at 99% and up).

    Record(
        name='index hit rate',
        ratio=Decimal('0.99994503346970922117')
    )

    :returns: list of Records
    """

    return self.execute(sql.CACHE_HIT)
```

## .calls(truncate=False)

```
def calls(self, truncate=False):
    """
    Show 10 most frequently called queries. Requires the pg_stat_statements
    Postgres module to be installed.

    Record(
        query='BEGIN;',
        exec_time=datetime.timedelta(0, 0, 288174),
        prop_exec_time='0.0%',
        ncalls='845590',
        sync_io_time=datetime.timedelta(0)
    )
```

```
        :param truncate: trim the Record.query output if greater than 40 chars
        :returns: list of Records
        """

        if self.pg_stat_statement():
            if truncate:
                select = """
                    SELECT CASE
                        WHEN length(query) < 40
                        THEN query
                        ELSE substr(query, 0, 38) || '..'
                    END AS qry,
                """
            else:
                select = 'SELECT query,'

            return self.execute(sql.CALLS.format(select=select))
        else:
            return [self.get_missing_pg_stat_statement_error()]
```

## .index_usage()

```
    def index_usage(self):
        """
        Calculates your index hit rate (effective databases are at 99% and up).

        Record(
            relname='pgbench_history',
            percent_of_times_index_used=None,
            rows_in_table=249976
        )

        :returns: list of Records
        """

        return self.execute(sql.INDEX_USAGE)
```

## .locks()

```
    def locks(self):
        """
        Display queries with active locks.

        Record(
            procpid=31776,
            relname=None,
            transactionid=None,
            granted=True,
            query_snippet='select * from hello;',
            age=datetime.timedelta(0, 0, 288174),
        )

        :returns: list of Records
        """
```

```
        return self.execute(
            sql.LOCKS.format(
                pid_column=self.pid_column,
                query_column=self.query_column
            )
        )
```

## .long_running_queries()

```
    def long_running_queries(self):
        """
        Show all queries longer than five minutes by descending duration.

        Record(
            pid=19578,
            duration=datetime.timedelta(0, 19944, 993099),
            query='SELECT * FROM students'
        )

        :returns: list of Records
        """

        if self.is_pg_at_least_nine_two():
            idle = "AND state <> 'idle'"
        else:
            idle = "AND current_query <> '<IDLE>'"

        return self.execute(
            sql.LONG_RUNNING_QUERIES.format(
                pid_column=self.pid_column,
                query_column=self.query_column,
                idle=idle
            )
        )
```

## .outliers()

```
    def outliers(self, truncate=False):
        """
        Show 10 queries that have longest execution time in aggregate. Requires
        the pg_stat_statments Postgres module to be installed.

        Record(
            qry='UPDATE pgbench_tellers SET tbalance = tbalance + ?;',
            exec_time=datetime.timedelta(0, 19944, 993099),
            prop_exec_time='67.1%',
            ncalls='845589',
            sync_io_time=datetime.timedelta(0)
        )

        :param truncate: trim the Record.qry output if greater than 40 chars
        :returns: list of Records
        """
```

```
        if self.pg_stat_statement():
            if truncate:
                query = """
                    CASE WHEN length(query) < 40
                        THEN query
                        ELSE substr(query, 0, 38) || '..'
                    END
                """
            else:
                query = 'query'

            return self.execute(sql.OUTLIERS.format(query=query))
        else:
            return [self.get_missing_pg_stat_statement_error()]
```

## .ps()

```
    def ps(self):
        """
        View active queries with execution time.

        Record(
            pid=28023,
            source='pgbench',
            running_for=datetime.timedelta(0, 0, 288174),
            waiting=0,
            query='UPDATE pgbench_accounts SET abalance = abalance + 423;'
        )

        :returns: list of Records
        """

        if self.is_pg_at_least_nine_two():
            idle = "AND state <> 'idle'"
        else:
            idle = "AND current_query <> '<IDLE>'"

        return self.execute(
            sql.PS.format(
                pid_column=self.pid_column,
                query_column=self.query_column,
                idle=idle
            )
        )
```

## .seq_scans()

```
    def seq_scans(self):
        """
        Show the count of sequential scans by table descending by order.

        Record(
            name='pgbench_branches',
            count=237
        )
```

```
        :returns: list of Records
        """

        return self.execute(sql.SEQ_SCANS)
```

## .total_table_size()

```
    def total_table_size(self):
        """
        Show the size of the tables (including indexes), descending by size.

        Record(
            name='pgbench_accounts',
            size='15 MB'
        )

        :returns: list of Records
        """

        return self.execute(sql.TOTAL_TABLE_SIZE)
```

## .unused_indexes()

```
    def unused_indexes(self):
        """
        Show unused and almost unused indexes, ordered by their size relative
        to the number of index scans. Exclude indexes of very small tables
        (less than 5 pages), where the planner will almost invariably select
        a sequential scan, but may not in the future as the table grows.

        Record(
            table='public.grade_levels',
            index='index_placement_attempts_on_grade_level_id',
            index_size='97 MB',
            index_scans=0
        )

        :returns: list of Records
        """

        return self.execute(sql.UNUSED_INDEXES)
```

## .vacuum_stats()

```
    def vacuum_stats(self):
        """
        Show dead rows and whether an automatic vacuum is expected to be
        triggered.

        Record(
            schema='public',
            table='pgbench_tellers',
            last_vacuum='2014-04-29 14:45',
```

```
        last_autovacuum='2014-04-29 14:45',
        rowcount='10',
        dead_rowcount='0',
        autovacuum_threshold='52',
        expect_autovacuum=None
    )

    :returns: list of Records
    """

    return self.execute(sql.VACUUM_STATS)
```

## .version()

```
def version(self):
    """
    Get the Postgres server version.

    Record(
        version='PostgreSQL 9.3.3 on x86_64-apple-darwin13.0.0'
    )

    :returns: list of Records
    """

    return self.execute(sql.VERSION)
```

CHAPTER 4

# Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

## Types of Contributions

### Report Bugs

Report bugs at https://github.com/scottwoodall/pgextras/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

### Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

### Write Documentation

python-pgextras could always use more documentation, whether as part of the official python-pgextras docs, in docstrings, or even on the web in blog posts, articles, and such.

### Submit Feedback

The best way to send feedback is to file an issue at https://github.com/scottwoodall/pgextras/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## Get Started!

Ready to contribute? Here's how to set up *pgextras* for local development.

1. Fork the pgextras repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/python-pgextras.git
   ```

3. Create a Postgres database with the following name and load it with test data:

   ```
   $ psql -c 'CREATE database python_pgextras_unittest'
   $ make populate-test-db
   ```

4. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv pgextras
   $ cd python-pgextras/
   $ python setup.py develop
   $ pip install -r requirements.txt
   ```

5. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

6. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ make lint
   $ make test
   $ make test-all
   ```

7. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

8. Submit a pull request through the GitHub website.

## Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.

3. The pull request should work for Python 2.7, and 3.3. Check https://travis-ci.org/scottwoodall/pgextras/pull_requests and make sure that the tests pass for all supported Python versions.

## Tips

To run a subset of tests:

```
$ python -m unittest tests.test_pgextras
```

CHAPTER 5

# Credits

## Development Lead

- Scott Woodall <scott.woodall@gmail.com>

## Contributors

None yet. Why not be the first?

CHAPTER 6

# History

## 0.2.1 (2018-12-01)

- Fixed bug that was truncating index names to only 63 characters
- Updated various sql statement to match heroku's version (thanks @mantrala)

## 0.2.0 (2014-05-06)

- Added a CLI option. Original idea for this came from Marek Wywiał.
- Remove commas from integer output so no additional parsing is needed if someone wants to store the results.
- Make API more consistent by not raising an exception when the "pg_stat_statement" module is not installed. Instead a list with a single namedtuple containing the error message is returned.

## 0.1.1 (2014-05-01)

- updated documentation
- increased test coverage

## 0.1.0 (2014-04-25)

- First release on PyPI.

CHAPTER 7

# Indices and tables

- *genindex*
- *modindex*
- *search*