“psycopg”

all about the love
between Python and PostgreSQL

‘2021-11-17’::date

PostgreSQL

- You know what we are talking about
- (it’s a database)

Python

- High level programming language
- Ubiquitous
- The one with the whitespaces

https://github.com/psycopg/psycopg/blob/master/psycopg/psycopg/connec 120%

```
448         yield from self._exec_command(self._get_tx_start_command())
449
450     def _get_tx_start_command(self) -> bytes:
451         if self._begin_statement:
452             return self._begin_statement
453
454         parts = [b"BEGIN"]
455
456         if self.isolation_level is not None:
457             val = IsolationLevel(self.isolation_level)
458             parts.append(b"ISOLATION LEVEL")
459             parts.append(val.name.replace("_", " ").encode())
460
461         if self.read_only is not None:
462             parts.append(b"READ ONLY" if self.read_only else b"READ WRITE")
463
464         if self.deferrable is not None:
465             parts.append(
466                 b"DEFERRABLE" if self.deferrable else b"NOT DEFERRABLE"
467             )
468
469         self._begin_statement = b" ".join(parts)
470         return self._begin_statement
```

Psycopg!

psycopg2:
a long history

```
commit c89f82112604e6235b1822e4ad0d619c777d4cfb
Author: Federico Di Gregorio <fog@initd.org>
Date:   Fri Oct 29 16:08:31 2004 +0000

    SVN repo up to date (1.1.16pre1).

commit c904d97f696a665958c2cc43333d09c0e6357577
Author: Federico Di Gregorio <fog@initd.org>
Date:   Tue Oct 19 03:17:12 2004 +0000

    Initial psycopg 2 import after SVN crash.
(END)
```

psycopg **3:**
a new story

```
commit aecc520b010bc58f86c90ef9c39bc86a3343aa73
Author: Daniele Varrazzo <daniele.varrazzo@gmail.com>
Date:   Fri Mar 13 23:27:19 2020 +1300

    Adding a first implementation of the libpq wrapper and tests

commit 45146ebffb4c23439c56cd2075ff0bd5925b9d43
Author: Daniele Varrazzo <daniele.varrazzo@gmail.com>
Date:   Sat Mar 7 16:56:16 2020 +0000

    The first commit is a thank you
(END)
```

Why psycopg 3?

- Introducing server-side parameters binding
  - breaking change
- Pure Python + C speedup
  - to allow non cpython use
- Async I/O support
- Fixing the last 15 years of mistakes
  - Preparing the next mistakes...

Psycopg 3:
A sponsored project!

Postgres Pro

Command Prompt

Dalibo

sponsors >

What is Psycopg?

- ⚙️ industry standard Python-PostgreSQL adapter
- 📚 it's a library (not a framework ⚙️ not a daemon 👻 ...)
- 💪 libpq based
- ⚖️ part% Python, (1 - part%) C
- (🐍 + 🅒) + 🐘 = ❤️

psycopg 2 docs >
psycopg 3 docs >

Installation

- **`pip install psycopg`**
  - requires system libpq
- **`pip install psycopg[c]`**
  - requires C compiler and -dev libraries
  - requires system libpq
- **`pip install psycopg[binary]`**
  - self-contained binary packages
  - on supported platforms (now on Alpine Linux too!)

install docs >

Usage

```
>>> import psycopg
>>> with psycopg.connect(“dbname=piro”) as conn:
...     cur = conn.execute(“SELECT * FROM mytable”)
...     cur.fetchone()
(“hello, world”, 42)
```

usage docs >

Passing parameters

```
# The bad way...
>>> conn.execute(
...     “INSERT INTO mytable VALUES (‘%s’, ‘%s’)” %
...     (value1, value2))
# Don’t do this! 💀
```

params docs >

Passing parameters

```
# The good way!
>>> conn.execute(
...     “INSERT INTO mytable VALUES (%s, %s)”,
...     (value1, value2))
# psycopg takes care of the translation 👍
```

params docs >

Receiving results

```
>>> cursor.fetchone()          # -> record
>>> cursor.fetchmany(n)        # -> list of records
>>> cursor.fetchall()          # -> list of records
>>> for record in cursor:
...     do_something(record)   # -> iteration
```

params docs >

What is a record?

```
# Basic tuples
(“John”, 42)
# Dictionaries
{“name”: “John”, “age”: 42}
# Any custom object
Person(name=“John”, age=42)
```

row factories >

Native data types

- Strings (binary, unicode)
- Numbers (integer, fixed precision, floating point)
- Date/time objects, timezones
- Arrays
- Network types
- UUID
- JSON!
- ...
- (XML is left as exercise)

data types >

PostgreSQL data types

- Composite
- Range
- Multirange (from v14)
- Hstore
- Geometric types

extension types >

Transactions control

```
>>> with conn.transaction():
...     conn.execute(“INSERT 1”)
...     conn.execute(“INSERT 2”)
...     with conn.transaction() as tx:
...         conn.execute(“nested”)
...         raise Rollback(tx)
```

transactions >

COPY support

```
>>> with cur.copy("COPY mytable FROM STDIN") as copy:
...     for row in some_generator():
...         copy.write_row(row)
>>> with cur.copy("COPY mytable TO STDOUT") as copy:
...     for row in copy.rows():
...         some_consumer(row)
```

COPY >

Notifications

```
>>> conn.execute(“LISTEN mychan”)
>>> for notify in conn.notifies():
...     print(notify)
Notify(channel='mychan', payload='hello there')
```

Meanwhile in psql...

```
=# NOTIFY mychan, ‘hello there’;
```

notifications >

Connection pools

```
>>> with ConnectionPool() as pool:
...     # bring it on, threads
```

- Lightweight, in-process connection pool
- Not a replacement for pgbouncer (out-of-process)

connection pools >

...and more

- async or multi-thread communication
- support for static typing
- prepared statements
- binary data support
- libpq access from Python

What’s next?

- Batch/pipeline mode
- Streaming query (not supported yet by PostgreSQL)
- Logical replication (in psycopg2)
- JSON binary mode (under exploration)

Thank You!

questions?