Frequently Asked Questions

Table of contents

Questions

1. Project Questions

1.1. Why are there so many packages?

pg/python has many projects. Each are segregated into their own logical area. This is done to provide a clear distinction so that use of individual packages can be made without binding it to another. This allows other projects to conveniently depend on single packages, and allows package maintainers to construct the installation and update plan based on the individual packages.

1.2. Why are code names used so frequently?

Use of code names help to establish a major version of a project. For instance, one backend implementation is TEOP. If the project were to steer in a substantially different direction, a new code name would be appointed instead of clobbering the current practices. This allows for more natural version traversal.

1.3. How can I help out?

Experiment, patch, feedback. Just dive into the code. Give feedback, write patches, and experiment with it. If you know Python, the frontend and typical areas shouldn't be too difficult. However, the backend is a different story as it deals with both the PostgreSQL and Python C APIs. There is no easy entrance here.

2. Interface Protocol

2.1. What's GreenTrunk?

GreenTrunk is the code name of an interface protocol, a Python API specification. It formally specifies the expected features of a conforming interface. See the Interface Protocol project for more information.

2.2. Why not use the DB-API 2.0 standard?

DB-API 2.0 specifies a decent interface, but it fails to provide a very Pythonic interface for

PostgreSQL. The Interface Protocol attempts to create a Python interface that objectifies PostgreSQL elements, something that DB-API could not have done due to its broader application. Nonetheless, it is likely that all interface implementations will include DB-API 2.0 layers.

3. Frontend

3.1. What's Proboscis?

Proboscis is the code name of the frontend implementation. See the project page for more information.

3.2. Why yet another %$#%@#@ driver?

Firstly, watch yer language. Secondly, the answer is flexibility. No other driver provides the level of flexibility that pg_pqueue and pg_proboscis do. pg_proboscis can work over an arbitrary data stream(pipe keyword), and may be useable in systems like the Twisted framework without using threads.

3.3. How slow is it?

The primary bottleneck on cursor fetches is the local typing(parsing or reading the individual object data and creating the Python object to represent it). This means that the protocol parts of the system are not a substantial impediment on performance. It should be able to reach speeds close to the libpq wrappers.

4. Backend

4.1. What's pg_teop?

pg_teop is the name of the backend interface. It is an acronym and stands for The Elusive Ophidian Proboscidea.

4.2. Why do I get this pthread symbol not found error?

On FreeBSD, PostgreSQL needs to be compiled against libc_r, in 4.x, or libpthread, in 5.x to work with Python. You either need to recompile PostgreSQL against the appropriate library, or execute PostgreSQL with the LD_PRELOAD environment variable set to the path of the

appropriate threading library stated earlier(/usr/lib/libc_r.so or /usr/lib/libpthread.so).