Digital Career Institute

Python Course - Databases - 3.3 Usage in Python

Digital Career Institute

Mapping between Python and Postgres Types

Python and Postgres Types

There is default mapping specified to convert Python types into PostgreSQL equivalent, and vice versa.

Whenever you execute a PostgreSQL query using Python table[1 in the next slide] is used by psycopg2 to return the result in the form of Python objects.

[1] https://www.postgresql.org/docs/9.4/plpython-data.html

Python and Postgres Types

Python	PostgreSQL
None	NULL
bool	bool
float	real or double
int	smallint integer bigint
Decimal	numeric
str	varchar text
date	date
time	time timetz

Python and Postgres Types

Digital Career Institute

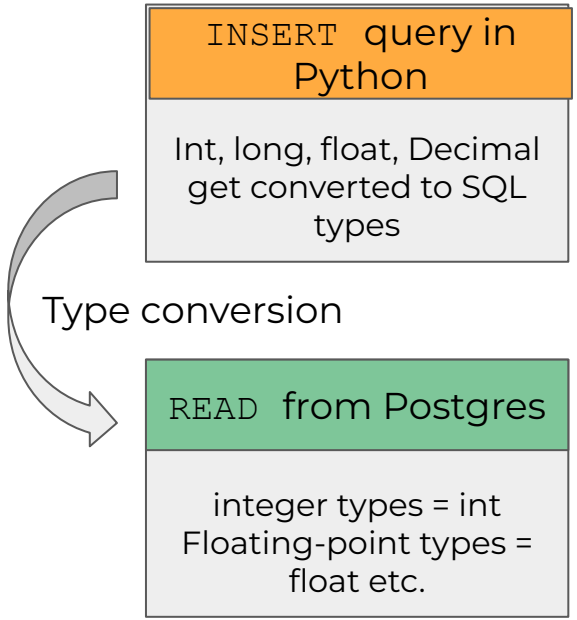
datetime	timestamp timestamptz
timedelta	interval
list	ARRAY
tuple	Composite types IN syntax
dict	hstore

Constant and Numeric Conversions

When you try to insert Python `None` and `boolean` values into Postgres, the respective value gets converted into the proper SQL literals. The same is true with Python numeric types. Numeric types get converted into equivalent PostgreSQL types.

⚠️

Example



Psycopg2 Exceptions

Psycopg2 `Exceptions` Module

- Psycopg2 comes with a few built-in exceptions to help debug programs [1].

- Two most commonly occurring exceptions in psycopg2 are `OperationalError` and `ProgrammingError` exception classes.

[1] https://www.postgresql.org/docs/9.2/errcodes-appendix.html

Psycopg2 `Exceptions` Module

Digital Career Institute

An **`OperationalError`** typically occurs when the parameters passed to the **`connect()`** method are incorrect, or if the server runs out of memory, or if a piece of datum cannot be found, etc.

A **`ProgrammingError`** happens when there is a syntax error in the SQL statement string passed to the **`execute()`** method, or if a SQL statement is executed to delete a non-existent table, or an attempt is made to create a table that already exists.

Other Psycopg2 `Exceptions`

Digital Career Institute

Exception Class	Function
`InterfaceError`	Raised for errors that are related to the database interface rather than the database itself.
`DatabaseError`	Raised for errors that are related to the database.
`DataError`	Raised for errors that are due to problems with the processed data like division by zero, numeric value out of range, etc.
`OperationalError`	Raised for errors that are related to the database's operation and not necessarily under the control of the programmer.
`IntegrityError`	Raised when the relational integrity of the database is affected.
`InternalError`	Raised when the database encounters an internal error.
`ProgrammingError`	Raised for programming errors.
`NotSupportedError`	Raised in case a method or db API was used which is not supported by the database.

Psycopg2 `Exceptions` Hierarchy

Digital Career Institute

```
StandardError
|__ Warning
|__ Error
     |__ InterfaceError
     |__ DatabaseError
          |__ DataError
          |__ OperationalError
          |__ IntegrityError
          |__ InternalError
          |__ ProgrammingError
          |__ NotSupportedError
```

At the core of the lesson

- Postgres can be used with Python using adapters.
- Psycopg2 is the most common and stable adapter.
- Psycopg2 allows to write SQL queries right in the Python code.
- CRUD operations are the backbone of a persistent application.

Connection Pooling

What is Connection Pool

Digital Career Institute

Postgres connection pools are **cached database connections created and maintained to get reused** for incoming requests instead of making the new connection every time.

The primary benefits of connection pooling are **time and performance improvements**.

What is Connection Pool

Digital Career Institute

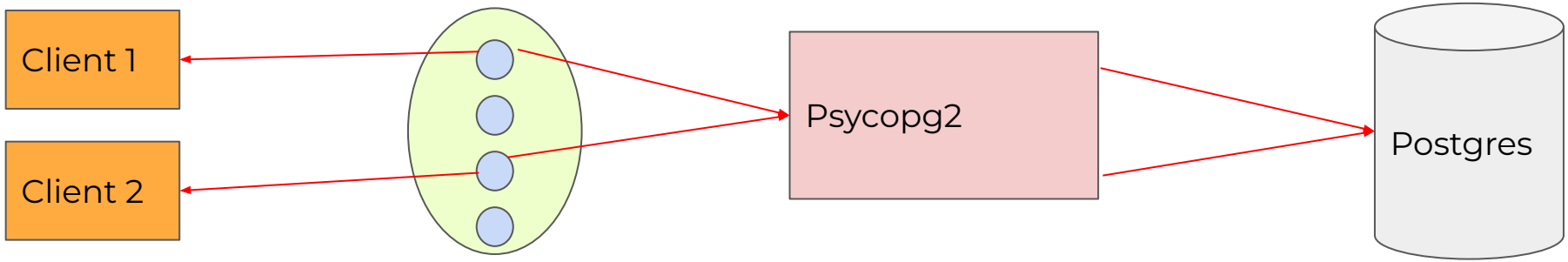
Analogy - Think of your connections like a delivery service:

If you are delivering a hundred packages, it's going to take a long time and result in a lot of wasted effort.

It's better for you and the courier if you can answer the door once, and sign for as many parcels as you feel like carrying upstairs before returning for some more.

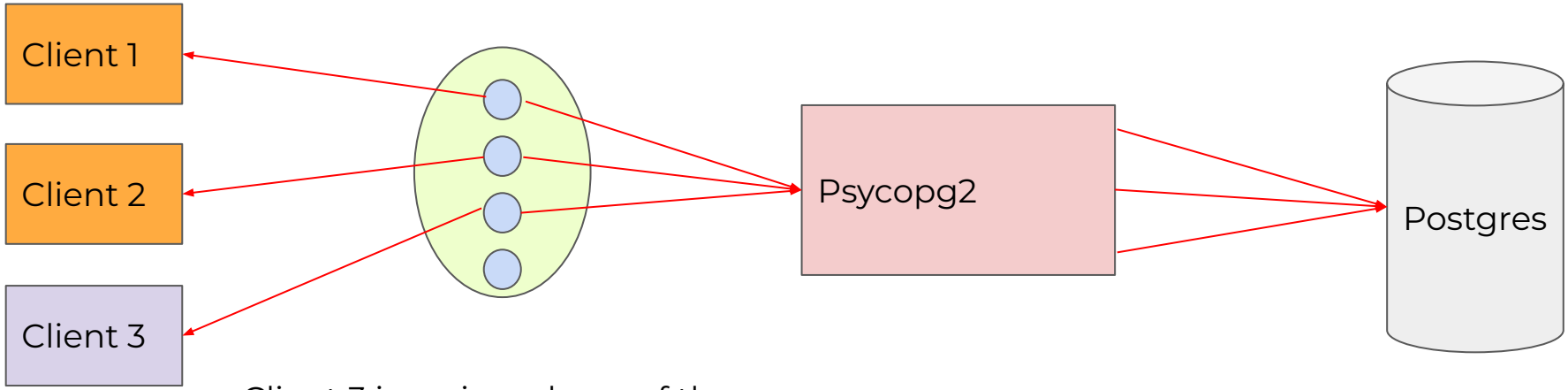
In a nutshell, that is what **connection pooling** offers.

Connection Pooling Visualisation



Connection pool of 4 instances, with 2 open connections.

Connection Pooling Visualisation



Client 3 is assigned one of the open connections pools, instead of establishing a new connection to DB.

Psycopg2's Connection Pool Classes

The Psycopg2 library has the following four classes to manage the Postgres connection pool:

- `AbstractConnectionPool`
- `SimpleConnectionPool`
- `ThreadedConnectionPool`
- `PersistentConnectionPool`

AbstractConnectionPool

Digital Career Institute

It is a base class implementing generic key-based pooling code.

An `AbstractConnectionPool` is an abstract class.

```
psycopg2.pool.AbstractConnectionPool(minConnection, maxConnection, *args, **kwargs)
```

- `minConnection`: Minimum connection objects required.
- `*args`, `**kwargs`, `min and maxConnections` are the necessary arguments for a `connect()` method to establish a connection to the Postgres database.

AbstractConnectionPool

Digital Career Institute

It is a subclass of the `AbstractConnectionPool` class and implements methods defined in it. It is ready to use class for the connection pool.

This class is suitable only for single-threaded application. If a connection pool using is created using this class then this pool cannot be shared across different threads.

```
psycopg2.pool.SimpleConnectionPool(minConnection, maxConnection, *args, **kwargs)
```

- `minConnection`: Minimum connection objects required.
- `*args`, `**kwargs`, `min and maxConnections` are the necessary arguments for a `connect()` method to establish a connection to the Postgres database.

ThreadedConnectionPool

Digital Career Institute

It is also a subclass of the `ThreadedConnectionPool` class and implements methods defined in it. Ready to use for the connection pool.

As the name suggests, **this class used in a multithreaded environment**.

```
psycopg2.pool.ThreadedConnectionPool(minConnection, maxConnection, *args,
**kwargs)
```

- `minConnection`: Minimum connection objects required.
- `*args`, `**kwargs`, `min and maxConnections` are the necessary arguments for a `connect()` method to establish a connection to the Postgres database.

PersistentConnectionPool

Digital Career Institute

It is also a subclass of the `PresistentConnectionPool` class and implements methods defined in it.

As the name suggests, each thread gets a single connection from the pool. The thread can't use more than one connection from the pool.

```
psycopg2.pool.PersistentConnectionPool(minConnection, maxConnection, *args,
**kwargs)
```

- `minConnection`: Minimum connection objects required.
- `*args`, `**kwargs`, `min and maxConnections` are the necessary arguments for a `connect()` method to establish a connection to the Postgres database.

Postgres and Django

Django Migrations

Digital Career Institute

"Migrations are Django's way of propagating changes you make to your models (adding a field, deleting a model, etc.) into your database schema."
- *Django Docs*

Migrations are required in pretty much in every Django application.

Migrations mainly allow developers to:

- Define database schemas
- Perform changes (CRUD) to the databases
- Indexing the database

SQL and Django Problem

Digital Career Institute

- Previously, we learned that SQL is used to create a database schema.
- These have to be the same as models that need to be defined in Django applications.
- Writing SQL with the same constraints as models is difficult. It requires a decent knowledge of SQL.
- Django solved this problem with the introduction of Django Object-relational Mapper (ORM).

Benefits of Django Migrations

Digital Career Institute

- Create tables without writing SQL.
- Models and database schemas are always synchronised.
- Don’t repeat yourself (DRY) philosophy (since Django migrations automatically generates SQL).
- Easier version control for database scheme.

At the core of the lesson

- Connection pools allow caching of connections to a Postgres database.
- Pooling optimizes the performance of a persistent storage.
- In python, connections can be pooled using `AbstractConnectionPool`, `SimpleConnectionPool`, `ThreadedConnectionPool`, `PersistentConnectionPool`.
- In Django, all database related tasks can be managed using Migrations.

Documentation

Documentation

Digital Career Institute

1. Psycopg2 Docs
2. Postgres Docs
3. Django Migrations

Contact Details
DCI Digital Career Institute gGmbH

Digital Career Institute