**Setting up databases with PostgreSQL, PSequel, and Python**



(image source: <https://avataai.com/big-data-streaming-processing/>)

As the demand for Data Scientists [continues to increase](https://edgylabs.com/data-scientist-is-americas-best-career-for-the-third-year-running), and is being dubbed the the “sexiest job job of the 21st century” by various outlets (including [Harvard Business Review](https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century)), questions have been asked of what skills should aspiring data scientists master on their way to their first data analyst job.

There is now a plethora of online courses to gain the skills needed for a data scientist to be good at their job (excellent reviews of online resources [here](https://medium.com/personal-growth/16-top-rated-data-science-courses-e5e161b937c3) and [here](https://medium.freecodecamp.org/a-path-for-you-to-learn-analytics-and-data-skills-bd48ccde7325)). However, as I reviewed the various courses myself , I noticed that a lot of focus is put on exciting and flashy topics like Machine Learning and Deep Learning without covering the basics of what is needed to gather and store datasets needed for such analysis.

**(Big) Data Science**

Before we go into PostgreSQL, I suspect many of you have the same question: why should I care about SQL?

Although database management may seem like a boring topic for aspiring data scientists — implementing a dog breed classifier is very rewarding I know! — it is a necessary skill once you join the industry and the data supports this: SQL remains the most common and [in-demand skill](https://www.kdnuggets.com/2016/02/data-science-skills-2016.html) listed in LinkedIn job postings for data science jobs.

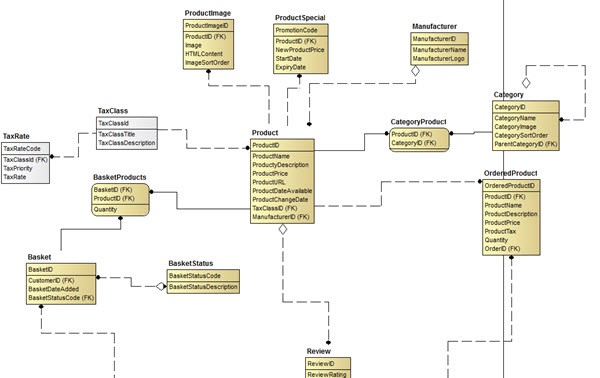


SQL is a necessary skills in many data science applications with large datasets (image source: c<http://exponential.io/blog/2015/02/26/big-data-sql/>)

[Pandas](https://pandas.pydata.org/) can perform the most common SQL operations well but is not suitable for large databases — its main limit comes to the amount of data one can fit in memory. Hence, if a data scientist is working with large databases, SQL is used to transform the data into something manageable for pandas before loading it in memory.

Furthermore, SQL is much more than just a method to have flat files dropped into a table. The power of SQL comes in the way that it allows users to have a set of tables that “relate” to one another, this is often represented in an “Entity Relationship Diagram” or ERD.

Many data scientists use both simultaneously — they use SQL queries to join, slice and load data into memory; then they do the bulk of the data analysis in Python using [pandas](https://pandas.pydata.org/) library functions.



Example of an ERD (source: <http://www.datanamic.com/dezign/erdiagramtool.html>)

This is particularly important when dealing with the large datasets found in Big Data applications. Such applications would have tens of TBs in databases with several billion rows.

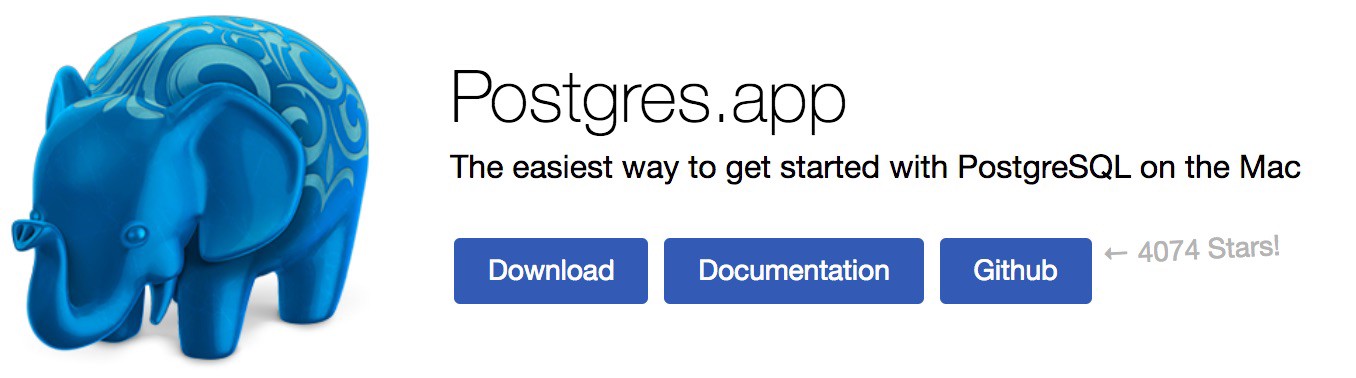
Data Scientist often start with SQL queries that would extract 1% of data needed to a csv file, before moving to Python Pandas for data analysis.

**Enter PostgreSQL**

There is a way to learn SQL without leaving the much loved Python environment in which so much Machine Learning and Deep Learning techniques are taught and used: PostgreSQL.

[PostgreSQL](https://www.postgresql.org/) allows you to leverage the amazing [Pandas library](https://pandas.pydata.org/) for data wrangling when dealing with large datasets that are not stored in flat files but rather in databases.

PostgreSQL can be installed in Windows, Mac, and Linux environments (see [install details here](https://www.tutorialspoint.com/postgresql/postgresql_environment.htm)). If you have a Mac, I’d highly recommend installing the [Postgres.App](http://postgresapp.com/)SQL environment. For Windows, check out [BigSQL](https://www.openscg.com/bigsql/postgresql/installers.jsp/).



PostgreSQL uses a client/server model. This involves two running processes:

* **Server process**: manages database files, accepts connections to the database from client applications, and performs database actions on behalf of the clients.
* **User Client App**: often involves SQL command entries, a friendly graphical interface, and some database maintenance tool.

In real-case scenarios, the client and the server will often be on different hosts and they would communicate over a TCP/IP network connection.

**Installing Postgres.App and PSequel**

I’ll be focusing on Postgres.App for Mac OS in the rest of the tutorial.   
After installing [PostgresApp](https://postgresapp.com/), you can setup your first database by following the instructions below:

* Click “Initialize” to create a new server
* An optional step is to configure a$PATH to be able to use the command line tools delivered with Postgres.app by executing the following command in Terminal and then close & reopen the window:  
  sudo mkdir -p /etc/paths.d &&  
  echo /Applications/Postgres.app/Contents/Versions/latest/bin | sudo tee /etc/paths.d/postgresapp

You now have a PostgreSQL server running on your Mac with default settings:  
Host: localhost, Port: 5432, Connection URL: posgresql://localhost

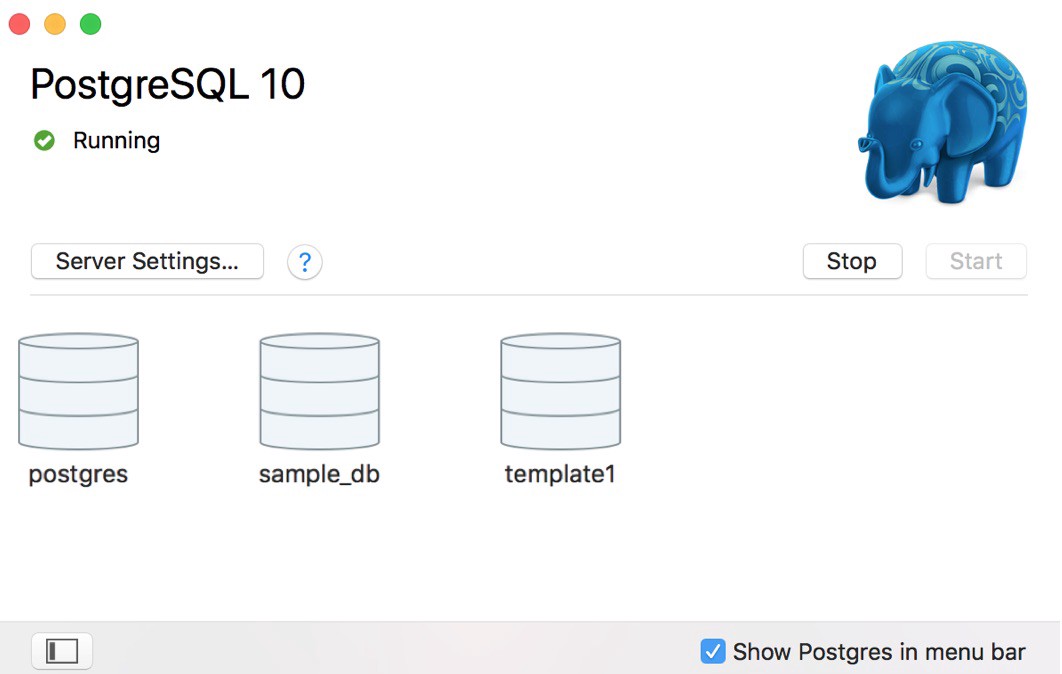
The PostgreSQL GUI client we’ll use in this tutorial is [PSequel](http://www.psequel.com/). It has a minimalist and easy to use interface that I really enjoy to easily perform PostgreSQL tasks.



Graphical SQL Client of choice: [PSequel](http://www.psequel.com/)

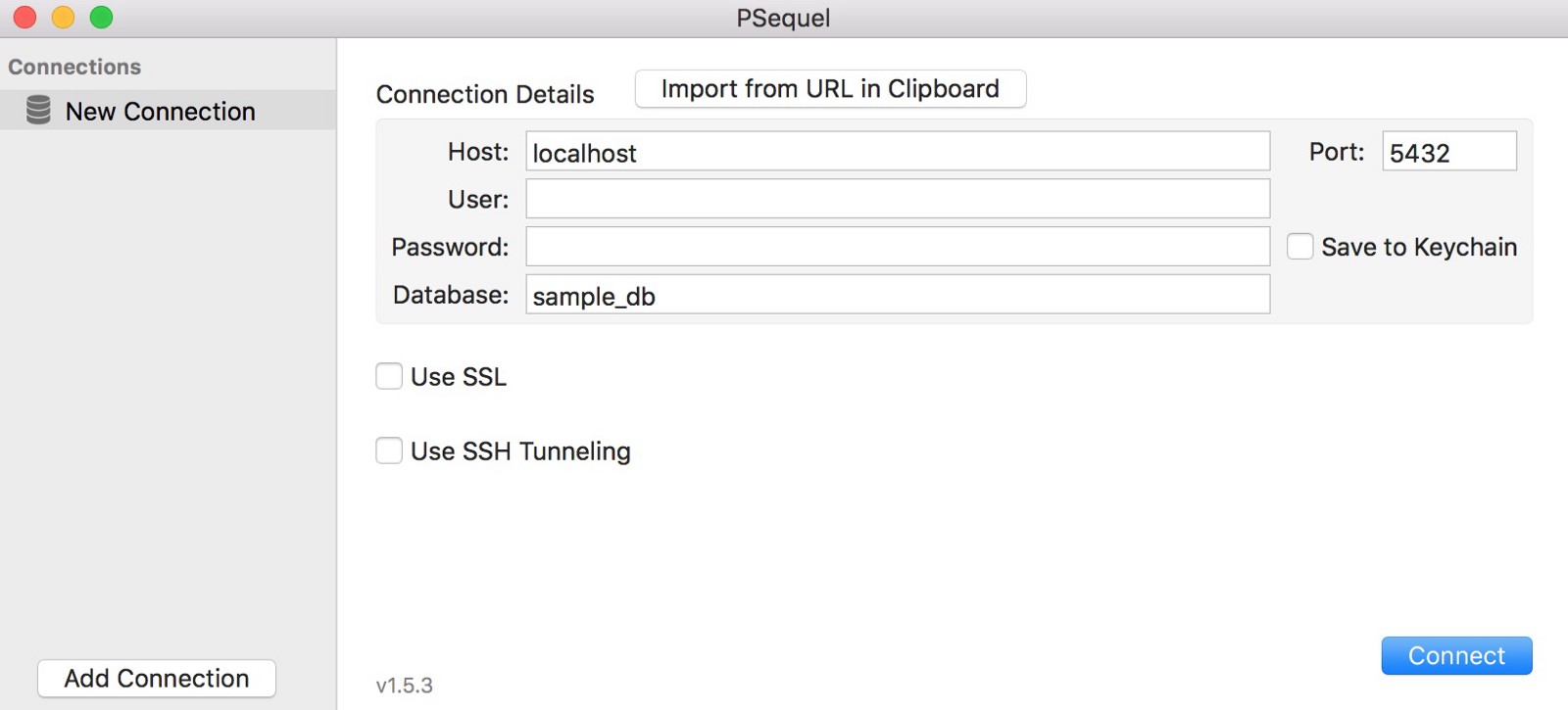
**Creating Your First Database**

Once Postgres.App and PSequel installed, you are now ready to set up your first database! First, start by opening Postgres.App and you’ll see a little elephant icon appear in the top menu.



You’ll also notice a button that allows you to “Open psql”. This will open a ommand line that will allow you to enter commands. This is mostly used to create databases, which we will create using the following command:  
create database sample\_db;

Then, we connect to the database we just created using PSequel. We’ll open PSequel and enter the name’s database, in our case: sample\_db . Click on “Connect” to connect to the database.



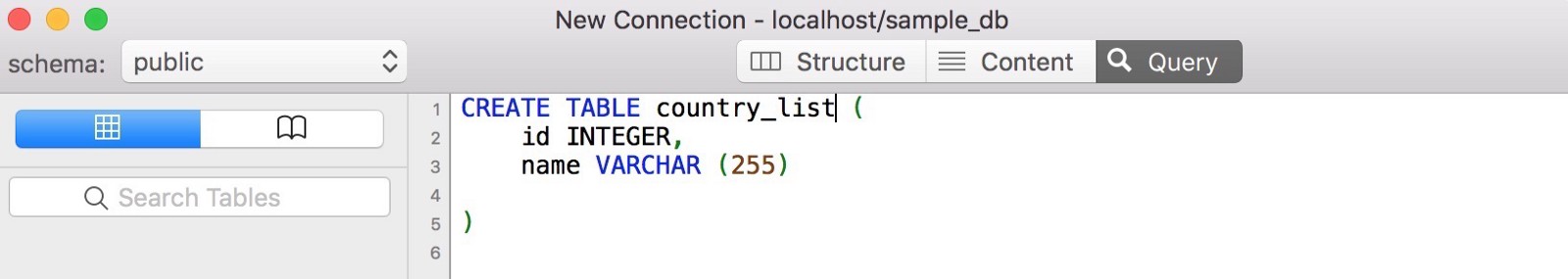
**Creating and Populating Your First Table**

Let’s create a table (consisting of rows and columns) in PSequel. We define the table’s name, and the name and type of each column.

The available datatypes in PostgreSQL for the columns (i.e. variables), can be found on the PostgreSQL [Datatypes Documentation](https://www.postgresql.org/docs/10/static/datatype.html).

In this tutorial, we’ll create a simple table with world countries. The first column will provide each country an ‘id’ integer, and the second column will provide the country’s name using variable length character string (with 255 characters max).

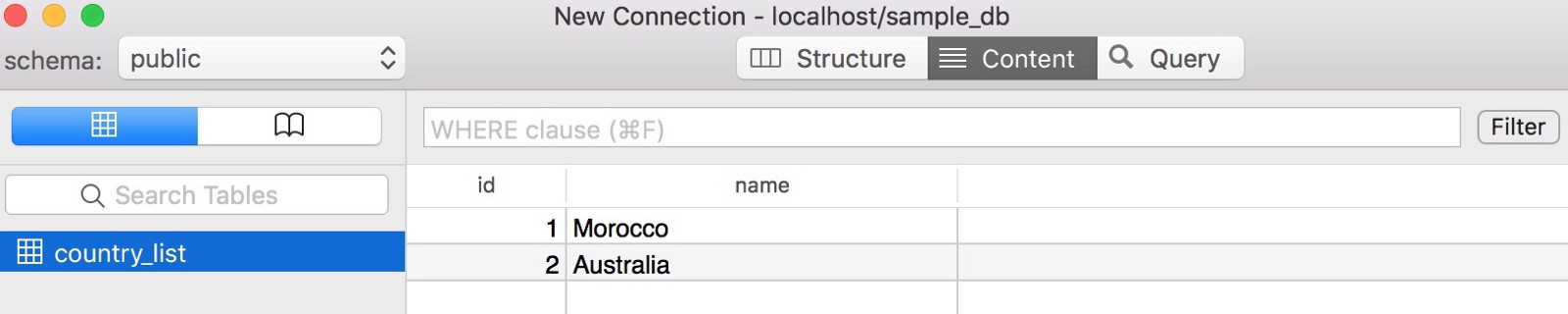
Once ready, click “Run Query”. The table will then be created in the database. Don’t forget to click the “Refresh” icon (bottom right) to see the table listed.



We are now ready to populate our columns with data. There are many different ways to populate a table in a database. To enter data manually, the INSERT will come in handy. For instance, to enter the country Morocco with id number 1, and Australia with id number 2, the SQL command is:

INSERT INTO country\_list (id, name) VALUES (1, 'Morocco');  
INSERT INTO country\_list (id, name) VALUES (2, 'Australia');

After running the query and refreshing the tables, we get the following table:



In practice, populating the tables in the database manually is not feasible. It is likely that the data of interest is stored in CSV files. To import a CSV file into the country\_list\_csv table, you use COPY statement as follows:

COPY country\_list\_csv(id,name)  
FROM 'C:\{path}\{file\_name}.csv' DELIMITER ',' CSV HEADER;

As you can see in the commands above, the table with column names is specified after the COPY command. The columns must be ordered in the same fashion as in the CSV file. The CSV file path is specified after the FROMkeyword. The CSVDELIMITER must also be specified.

If the CSV file contains a header line with column names, it is indicated with the HEADER keyword so that PostgreSQL ignores the first line when importing the data from the CSV file.

**Common SQL commands**

The key to SQL is understanding statements. A few statements include:

1. **CREATE TABLE** is a statement that creates a new table in a database.
2. **DROP TABLE** is a statement that removes a table in a database.
3. **SELECT** allows you to read data and display it.

**SELECT** is where you tell the query what columns you want back.  
**FROM** is where you tell the query what table you are querying from. Notice the columns need to exist in this table. For example, let’s say we have a table of orders with several columns but we are only interested in a subset of three:

SELECT id, account\_id, occurred\_at  
FROM orders

Also, the **LIMIT** statement is useful when you want to see just the first few rows of a table. This can be much faster for loading than if we load the entire dataset. The **ORDER BY** statement allows us to order our table by any row. We can use these two commands together to a table of ‘orders’ in a database as:

SELECT id, account\_id, total\_amt\_usd  
FROM orders  
ORDER BY total\_amt\_usd DESC  
LIMIT 5;

**Explore other SQL commands**

Now that you know how to setup a database, create a table and populate in PostgreSQL, you can explore the other common SQL commands as explained in the following tutorials:

* [SQL Basics for Data Science](https://towardsdatascience.com/how-to-ace-data-science-interviews-sql-b71de212e433)
* [Khan Academy: Intro to SQL](https://www.khanacademy.org/computing/computer-programming/sql/sql-basics/p/creating-a-table-and-inserting-data)
* [SQL in a Nutshell](https://towardsdatascience.com/sql-in-a-nutshell-part-1-basic-real-world-scenarios-33a25ba8d220)
* [Cheat Sheet SQL Commands](https://www.codecademy.com/articles/sql-commands)

**Accessing PostgreSQL database in Python**

Once your PostgreSQL database and tables are setup, you can then move to Python to perform any Data Analysis or Wrangling required.

PostgreSQL can be integrated with Python using [psycopg2 module](http://initd.org/psycopg/docs/). It is a popular PostgreSQL database adapter for Python. It is shipped along with default libraries in Python version 2.5.x onwards

Connecting to an existing PostgreSQL database can be achieved with:

import psycopg2  
conn = psycopg2.connect(database="sample\_db", user = "postgres", password = "pass123", host = "127.0.0.1", port = "5432")

Going back to our country\_list table example, inserting records into the table in sample\_db can be accomplished in Python with the following commands:

cur = conn.cursor()  
cur.execute("INSERT INTO country\_list (id, name) \  
 VALUES (1, 'Morocco')");  
cur.execute("INSERT INTO country\_list (id, name) \  
 VALUES (2, 'Australia')");  
conn.commit()  
conn.close()

Other commands for creating, populating and querying tables can be found on various tutorials on [Tutorial Points](https://www.tutorialspoint.com/postgresql/postgresql_python.htm) and [PostgreSQL Tutorial](http://www.postgresqltutorial.com/postgresql-python/).

**Conclusion**

You now have a working PostgreSQL database server ready for you to populate and play with. It’s powerful, flexible, free and is used by numerous applications.

# Menjadi Developer Web dengan Python dan Flask IV: Database



Artikel ini merupakan terjemahan [*The Flask Mega-Tutorial Part IV: Database*](https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-iv-database)dari seri The Flask Mega-Tutorial karya [*Miguel Grinberg*](https://blog.miguelgrinberg.com/) yang akan dirilis secara gratis bab per bab sampai bulan Mei 2018. Dukung penulis aslinya dengan membeli buku/video tutorial lengkapnya di [*sini*](https://learn.miguelgrinberg.com/).

Topik pembahasan bab ini sangat penting. Sebagian besar aplikasi akan membutuhkan suatu cara untuk menyimpan data yang dapat diambil secara efisien dan inilah kenapa database diciptakan.

The GitHub links for this chapter are: [*Browse*](https://github.com/miguelgrinberg/microblog/tree/v0.4), [*Zip*](https://github.com/miguelgrinberg/microblog/archive/v0.4.zip), [*Diff*](https://github.com/miguelgrinberg/microblog/compare/v0.3...v0.4).

## Database di Flask

Mungkin pembaca sudah tahu sebelumnya bahwa Flask tidak memiliki fitur database bawaan. Database merupakan salah satu area dimana kita diberikan kebebasan untuk memilih database apa yang cocok dengan aplikasi kita daripada dipaksa untuk menggunakan satu solusi saja.

Ada banyak pilihan database untuk Python yang bagus, kebanyaka mereka memiliki ekstensi Flask sehingga membuat proses integrasi ke aplikasi menjadi lebih baik. Database dapat dibagi menjadi dua grup yaitu grup yang mengikuti model relasi dan grup yang tidak mengikuti model relasi. Grup yang dikedua sering disebut NoSQL yang mengindikasikan bahwa mereka tidak mengimplementasi [SQL](https://en.wikipedia.org/wiki/SQL). Meski ada banyak produk database yang bagus di kedua grup tersebut, penulis merasa bahwa database relasiional lebih cocok untuk aplikasi yang memiliki struktur data seperti daftar user, blog, post, dll. sementara NoSQL cocok untuk data yang kurang terstruktur. Aplikasi yang akan kita buat dapat diimplementasi menggunakan database manapun, namun karena alasan yang sebelumnya sudah penulis kemukakan, maka kita hanya akan menggunakan database relasional.

Di artikel sebelumnya penulis sudah mengenalkan ekstensi Flask. Di bab ini kita akan kembai menggunakan dua ekstensi yaitu [Flask-SQLAlchemy](http://packages.python.org/Flask-SQLAlchemy), ekstensi Flask untuk menggunakan paket [SQLAlchemy](http://www.sqlalchemy.org/) yang sudah dibungkus sedemikian rupa sehingga lebih ramah terhadap kode Flask. SQLAlchemy adalah sebuah [Object Relational Mapper](http://en.wikipedia.org/wiki/Object-relational_mapping) atau ORM. ORM memungkinkan aplikasi untuk menggunakan database dengan data high-level seperti kelas, objek dan method daripada menggunakan tabel dan kode SQL secara langsung. Tugas ORM adalah menerjemahkan operasi high-level menjadi perintah database.

Hal yang menarik tentang SQLAlchemy adalah bahwa ia bukan ORM hanya untuk satu sistem database. SQLAlchemy mendukung banyak sistem database diantaranya [MySQL](https://www.mysql.com/), [PostgreSQL](https://www.postgresql.org/) dan [SQLite](https://www.sqlite.org/). Fitur ini sangat berguna karena kita bisa memakai sistem SQLite sederhana di lingkungan development lalu menggunakan sistem yang lebih terjamin untuk production menggunakan MySQL atau PostgreSQL tanpa mengubah aplikasi.

Untuk memasang Flask-SQLAlchemy, pastikan sudah mengaktifkan virtual environment (baca lagi artikel pertama jika lupa) dan jalankan:

(venv) $ pip install flask-sqlalchemy

## Database Migrations

Sebagian besar tutorial yang penulis temui membahas pembuatan dan penggunaan database namun tidak menunjukkan bagaimana menangani data yang sudah ada saat aplikasi semakin berkembang. Hal ini cukup berat karena database relasional fokus pada data yang terstruktur sehingga saat strukturnya berubah maka data yang sudah ada di database harus di migrasikan ke struktur yang baru.

Ekstensi kedua yang akan kita pakai di bab ini adalah [Flask-Migrate](https://github.com/miguelgrinberg/flask-migrate). Ekstensi ini merupakan pembungkus Flask untuk [Alembic](https://bitbucket.org/zzzeek/alembic), sebuah database migration framework-nya SQLAlchemy. Bekerja dengan database migrations akan menambah pekerjaan dalam menyiapkan database, namun tambahan kerjaan ini merupakan harga yang murah dibandingkan keuntungan yang akan kita dapatkan di masa mendatang.

Proses pemasangan Flask-Migrate mirip dengan proses yang sudah kita lakukan sebelumnya:

(venv) $ pip install flask-migrate

## Konfigurasi Flask-SQLAlchemy

Saat proses pembuatan aplikasi, penulis akan menggunakan SQLite database. SQLite database adalah pilihan paling nyaman untuk dipakai membuat aplikasi kecil, kadang aplikasi yang tidak begitu kecil pun masih bisa. Karena database disimpan di sebuah file maka kita tidak perlu menjalankan server database seperti MySQL dan PostgreSQL.

Kita akan menambah dua configuration baru kedalam config file:

import os

basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))

class Config(object):

# ...

SQLALCHEMY\_DATABASE\_URI = os.environ.get('DATABASE\_URL') or \

'sqlite:///' + os.path.join(basedir, 'app.db')

SQLALCHEMY\_TRACK\_MODIFICATIONS = False

Ekstensi Flask-SQLAlchemy mengambil lokasi file database melalui variabel SQLALCHEMY\_DATABASE\_URI . Seperti yang dapat pembaca ingat dari bab sebelumnya, mengambil konfigurasi dari environment variable dan menyiapkan nilai bawaan apabila environment variable tidak tersedia merupakan kebiasaan yang baik . Disini penulis mengambil alamat database dari environment variableDATABASE\_URL , dan apabila tidak tidak ada maka penulis akan membuat file app.db di direktori utama aplikasi ayng ditentukan oleh basedir.

Konfigurasi SQLALCHEMY\_TRACK\_MODIFICATIONS diatur nilainya menjadi Falseuntuk mendisable fitur Flask-SQLAlchemy yang tidak dibutuhkan (fitur untuk memberitahu aplikasi tiap kali terjadi perubahan di database)

Database dapat digunakan didalam aplikasi dalam bentuk database instance. Database migration engine uga akan memiliki sebuah instance. Berikut ini objek-objek yang perlu dibuat di file app/***init***.py :

from flask import Flask

from config import Config

from flask\_sqlalchemy import SQLAlchemy

from flask\_migrate import Migrate

app = Flask(\_\_name\_\_)

app.config.from\_object(Config)

db = SQLAlchemy(app)

migrate = Migrate(app, db)

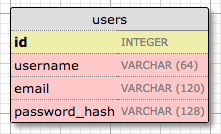
from app import routes, models

Ada tiga perubahan yang kita buat. Pertama, kita membuat objek db yang merepresentasikan database. Lalu kita juga menambah objek lain sebagai migration enginenya. Semoga pembaca sudah mulai bisa melihat pola penggunaan ekstensi Flask. Sebagian besar ekstensi diinisialisasikan seperti dua objek tersebut. Terakhir kita mengimpor modul baru bernama model di bagian bawah. Modul ini akan mendefinisikan struktur database.

## Database Models

Data yang akan disimpan didalam database diwakilkan oleh kumpulan kelas, biasanya disebut dengan database model. Lapiran ORM didalam SQLAlchem akan menerjemahkan objek dari model menjadi tabel-tabel SQL yang diperlukan.

Mari kita mulai membuat model yang mewakilkan user. Dengan memakai [WWW SQL Designer](http://ondras.zarovi.cz/sql/demo) tool, kita dapat membuat diagram seperti pada gambar:



Field id biasanya akan selalu ada disetiap model, dipakai sebagai primary key. Setiap user di database akan diberikan nilai id yang berbeda satu sama lain. Primary key pada umumnya akan selalu diisi secara otomatis oleh database, jadi kita cukup menandai id sebagai primary key.

Field username, email dan password\_hash diatur sebagai strings (atau VARCHARdalam istilah database), juga diberikan batas panjang agar database bisa mengoptimalkan penggunaan ruang yang dipakai. Field username dan emailmungkin sudah tidak perlu dijelaskan lagi fungsinya, namun password\_hashperlu mendapatkan perhatian lebih. Diaplikasi ini kita ingin agar saat database bocor, hacker tidak mendapatkan password dalam bentuk mentahnya melainkan dalam bentuk yang sudah diacak sehingga hacker tersebut akan mendapat kesulitan untuk menggunakannya. Untuk saat ini tidak perlu pusing memikirkannya, akan kita bahas lagi di masa mendatang.

Jadi sementara ini berikut tabel user yang akan kita pakai dan ditulis di file app/models.py:

from app import db

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(64), index=True, unique=True)

email = db.Column(db.String(120), index=True, unique=True)

password\_hash = db.Column(db.String(128))

def \_\_repr\_\_(self):

return '<User {}>'.format(self.username)

Kelas User diturunkan dari db.Model, base class untuk semua model Flask-SQLAlchemy. Kelas ini memiliki beberapa field sebagai variabel kelas. Field diisi dengan instance dari db.Column, dengan mengambil field sebagai salah satu argumennya, ditambah dengan argumen lain yang diperlukan misalnya menentukan field mana yang harus memiliki isi yang berbeda dan terindeks sehingga proses pencarian menadi lebih efisien.

Method \_\_repr\_\_ memberitahuu Python bagaimana mencetak objek kelas ini, berguna nanti untuk debugging. Pembaca dapat melihat cara kera \_\_repr\_\_() di interpreter Python di bawah:

>>> from app.models import User

>>> u = User(username='susan', email='susan@example.com')

>>> u

<User susan>

## Membuat Migration Repository

Kelas model yang dibuat di pembahasan sebelumnya menentukan struktur awal sebuah databas (atau dikenal juga dengan istilah schema). Saat aplikasi menjadi semakin besar, mungkin kita akan mengubah isinnya untuk menambah data baru juga mngubah serta menghapus data yang sudah ada. Alembic (migration framework yang dipakai Flask-Migrate) akan membuat perubahan schema sedemikian rupa sehingga database tidak perlu dibuat ulang dari awal.

Untuk mencapai tujuan ini Alembic menyimpan sebuah migration repository, yang merupakan direktori penyimpanan skrip migration. Setiap kali perubahan terjadi di schema database, sebuah skrip migration akan dieksekusi sesuai urutan dibuat.

Flask-Migrate menampilkan prosesnya dari perintah flask. Pembaca sudah melihat flask run, yang merupakan salah satu perintah asli Flask. Perintah flask db ditambah oleh Flask-Migrate untuk melakukan semua hal yang berhubungan dengan database migration. Jadi sekarang mari kita buat ulang migration repository untuk microblog dengan menajlankan flask db init:

(venv) $ flask db init

Creating directory /home/miguel/microblog/migrations ... done

Creating directory /home/miguel/microblog/migrations/versions ... done

Generating /home/miguel/microblog/migrations/alembic.ini ... done

Generating /home/miguel/microblog/migrations/env.py ... done

Generating /home/miguel/microblog/migrations/README ... done

Generating /home/miguel/microblog/migrations/script.py.mako ... done

Please edit configuration/connection/logging settings in

'/home/miguel/microblog/migrations/alembic.ini' before proceeding.

Ingat bahwa perintah flask bergantung pada environment variable FLASK\_APPuntuk mengetahui di mana Flask berada. Untuk aplikasi ini pembaca perlu mengatur FLASK\_APP=microblog.py, seperti yang sudah didiskusikan di bab 1.

Setelah menjalankan perintah ini pembaca akan menemukan direktori migrations yang baru dibuat dengan beberapa file dan direktori lainnya. Semua file ini harus menadi bagian dari aplikasi ini, dengan kata lain harus ditambahkan ke source control (contohnya git).

## Database Migration yang Pertama

Dengan adanya migration repository sekarang saatnya kita melakukan database migration. Ada dua cara untuk membuat database migration yaitu secara manual atau otomatis. Untuk membuat secara otomatis, Alembic membandingkan schema database yang ada di model dengan schema database yang sudah ada. Ia lalu membuat skrip migration untuk membuat perubahan yang diperlukan. Sekarang karena belum ada database yang pernah dibuat, maka migrasi otomatis akan menambah objek model User ke dalam skrip migration. Perintah flask db migrate akan membuat migrasi otomatis ini:

(venv) $ flask db migrate -m "users table"

INFO [alembic.runtime.migration] Context impl SQLiteImpl.

INFO [alembic.runtime.migration] Will assume non-transactional DDL.

INFO [alembic.autogenerate.compare] Detected added table 'user'

INFO [alembic.autogenerate.compare] Detected added index 'ix\_user\_email' on '['email']'

INFO [alembic.autogenerate.compare] Detected added index 'ix\_user\_username' on '['username']'

Generating /home/miguel/microblog/migrations/versions/e517276bb1c2\_users\_table.py ... done

Teks yang dihasilkan dari perintah di atas mungkin bisa menggambarkan apa yang ditambahkan olehh Alembic ke sebuah migration. Dua baris pertama biasanya bisa diabaikan. Alembic lalu mengatakan bahwa ia menemukan sebuah tabel user dan dua indeks. Ia lalu memberitahu dimana ia membuat skrip migrationnya. Kode e517276bb1c2 dibuat secara otomatis dan akan selalu berbeda untuk tiap orang. Opsi -m pada perintah di atas tidak wajib dipakai, ia hanya menambah deskripsi singkat disebelah baris migration.

Skrip migration yang dibuat juga menjadi bagian dari aplikasi sehingga perlu ditambahkan juga ke source control. Pembaca bisa melihat isi skrip tersebut jika penasaran dengan isinya. Pembaca akan melihat dua fungsi bernama upgrade()dan downgrade(). Fungsi upgrade() akan mengaplikasikan migration dan downgrade() akan menghapusnya (kebalikan). Kedua fungsi ini memungkinkan Alembic untuk mengatur database untuk kembali ke schema tertentu dari versi migration sebelum-sebelumnya dengan fungsi downgrade.

Perintah flask db migrate tidak mengubah apapun ke database, ia hanya membuatkan skrip migration. Untuk melakukan perubahan di database gunakan perintah flask db upgrade.

(venv) $ flask db upgrade

INFO [alembic.runtime.migration] Context impl SQLiteImpl.

INFO [alembic.runtime.migration] Will assume non-transactional DDL.

INFO [alembic.runtime.migration] Running upgrade -> e517276bb1c2, users table

Karena aplikasi ini menggunakan SQLite, perintah upgrade akan mendeteksi abhwa sebuah database belum ada sehingga ia akan membuatnya (akan ada file database SQLite baru bernama app.db setelah perintah sukses). Jika menggunakan server database seperti MySQL dan PostgreSQL, kita harus membuat dulu sebuah database di server sebelum menjalankan upgrade.

## Alur Kerja Database Upgrade dan Downgrade

Meski belum bisa apa-apa, tidak salah bila kita coba membahas sebentar bagaimana strategi database migration yang akan dilakukan nantinya. Bayangkan kita memiliki aplikasi di komputer development juga ada salinan di server yang sedang berjalan.

Anggap untuk rilis update berikutnya kita membuat perubahan di model, contohnya membuat sebuah tabel baru. Tanpa migrasi, kita harus menentukan sendiri bagaimana mengubah schema database, baik di komputer development juga di server sehingga ada banyak pekerjaan yang harus dilakukan.

Dengan menggunakan database migration, setelah memodifikasi model, aplikasi kita membuat skrip migration-nya (flask db migrate), lalu kita review untuk memastikan pembuatan migration otomatis sudah melakukan hal yang sesuai dan mengaplikasikan migration tadi ke database (flask db upgrade). Lalu kita tambahkan skrip migration ke source control dan meng-commit-nya.

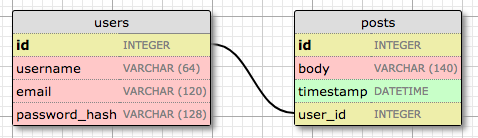
Setelah sudah siap untuk merilis versi baru aplikasi ke server production, yang perlu kita lakukan selanjutnya adalah mengirim versi aplikasi yang baru (termasuk skrip migration-nya) lalu menjalankan perintah flask db upgrade. Alembic akan mendeteksi bahwa database di production belum menggunakan schema terbaru sehingga akan menjalankan skrip-skrip baru setelah rilis sebelumnya.

## Relasi Database

Database relasional dapat menyimpan relasi antar data dengan sangat baik. Bayangkan kasus dimana seorang user menulis sebuah artikel blog. User tersebut akan memiliki data di tabel users dan artikel akan disimpan di tabel posts. Cara paling efektif untuk menyimpan data siapa yang menulis artikel tertentu ialah dengan menghubungkan dua tabel tersebut.

Setelah hubungan antara seorang user dan sebuah artikel terhubung, database dapat menjawab permintaan data yang diperlukan. Misalnya ada sebuah artikel dimana kita ingin mengetahui siapa user yang menulisnya, atau kita memiliki data seorang user dan ingin mencari semua artikel yang ditulisnya. Flask-SQLAlchemy dapat membantu kueri-kueri seperti ini.

Mari kita perbarui dataabse untuk menyimpan artikel (blog posts) untuk melihat bagaimana sebuah relasi terjadi. Berikut ini schema untuk tabel posts yang baru:



Tabel posts akan memiliki field id, body dan sebuah timestamp. Sebagai tambahan dari ketiga field tersebut, ada satu field tambahan bernama user\_idyang isinya adalah salah satu id di tabel users. Field user\_id disebut dengan foreign key. Hubungan seperti ini disebut dengan one-to-many karena "satu" (one) user dapat membuat "banyak" (many) posts atau artikel.

Kelas app/models.py yang sudah dimodifikasi terlihat sebagai berikut:

from datetime import datetime

from app import db

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(64), index=True, unique=True)

email = db.Column(db.String(120), index=True, unique=True)

password\_hash = db.Column(db.String(128))

posts = db.relationship('Post', backref='author', lazy='dynamic')

def \_\_repr\_\_(self):

return '<User {}>'.format(self.username)

class Post(db.Model):

id = db.Column(db.Integer, primary\_key=True)

body = db.Column(db.String(140))

timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)

user\_id = db.Column(db.Integer, db.ForeignKey('user.id'))

def \_\_repr\_\_(self):

return '<Post {}>'.format(self.body)

Kelas Post akan merepresentasikan artikel-artikel (blog posts) yang ditulis oleh user. Field timestamp akan di-index agar pengambilan data berdasarkan urutan menjadi lebih efisien. Kita juga menambahkan argumen default dan mengirimkan fungsi datetime.utcnow. Apa yang kita inginkan ialah mengambil waktu saat ini di zona UTC. Cara ini memastikan bahwa data waktu (timestamps) akan selalu sama dimanapun user berada. Data waktu yang disimpan dapat diubah menjadi waktu lokal saat ditampilkan.

Field user\_id diinisialisasi sebagai foreign key ke user.id, ini artinya kita akan memanggil nilai id dari tabel users. Kelas User memiliki field posts dan diisi dengan memanggil fungsi db.relationship. Dalam sebuah relasi one-to-many, db.relationship biasanya dianggap sebagai bagian "one", dan dipakai untuk mengambil akses bagian yang "many". Contohnya, jika kita memiliki user yang disimpan dalam objek u, maka pemanggilan u.posts akan meminta seluruh artikel yang ditulis oleh user. Argumen backref mendefinisikan nama field yang akan diberikan objek-objek bagian "many" atau bagian "one" yang akan memiliki banyak data artikel. Dengan begitu, saat memanggil post.author kita akan mendapatkan user yang menulis artikel tersebut. Argumen lazy mendefiniskan bagaimana kueri database untuk relasi dikirim, kita akan membahasnya nanti. Jangan risau jika penjelasan tadi belum begitu jelas, kita akan melihat contoh penerapannya diakhir artikel.

Karena kita membuat perubahan di model aplikasi, sebuah database migration perlu di buat:

(venv) $ flask db migrate -m "posts table"

INFO [alembic.runtime.migration] Context impl SQLiteImpl.

INFO [alembic.runtime.migration] Will assume non-transactional DDL.

INFO [alembic.autogenerate.compare] Detected added table 'post'

INFO [alembic.autogenerate.compare] Detected added index 'ix\_post\_timestamp' on '['timestamp']'

Generating /home/miguel/microblog/migrations/versions/780739b227a7\_posts\_table.py ... done

Lalu migration tadi harus diaplikasikan:

(venv) $ flask db upgrade

INFO [alembic.runtime.migration] Context impl SQLiteImpl.

INFO [alembic.runtime.migration] Will assume non-transactional DDL.

INFO [alembic.runtime.migration] Running upgrade e517276bb1c2 -> 780739b227a7, posts table

Jika menggunakan source control untuk menyimpan proyek ini, jangan lupa untuk menambahkan skrip ini kedalamnya.

## Play Time

Penulis sudah menyusahkan pembaca dalam proses panjang pembuatan database, namun penulis belum menunjukkan bagaimana kode-kode kita tadi bekerja. Sekarang mari kita bermain dengan database menggunakan interpreter Python untuk lebih mengenalnya. Maka, sekarang buka Python dengan memanggil perintah python. Pastikan virtual environment sudah aktif sebelum memanggilnya.

Setelah memasuki prompt Python, impor database instance dan model-modelnya:

>>> from app import db

>>> from app.models import User, Post

Mulai dengan membuat sebuah user:

>>> u = User(username='john', email='john@example.com')

>>> db.session.add(u)

>>> db.session.commit()

Perubahan ke database dilakukan dalam konteks session yang dapat diliat di db.session. Perubahan-perubahan yang dilakukan diakumulasi disebuah session dan setelah semua perubahan didaftarkan ke sana kita bisa mamanggil satu perintah db.session.commit() yang akan mengaplikasikan perubahan-perubahan tersebut. Jika saat bekerja dengan session terjadi error, db.session.rollback() akan dipanggil secara otomatis dan membatalkan semua perubahan di session tersebut. Yang perlu diingat adalah perubahan-perubahan hanya akan ditulis ke database hanya jika db.session.commit() di panggil.

Mari kita tambahkan user lain:

>>> u = User(username='susan', email='susan@example.com')

>>> db.session.add(u)

>>> db.session.commit()

Sistem dataabse dapat menjawab kueri yang mengembalikan semua user:

>>> users = User.query.all()

>>> users

[<User john>, <User susan>]

>>> for u in users:

... print(u.id, u.username)

...

1 john

2 susan

Semua model memiliki atribut query yang menjadi gerbang masuk ke kueri database lainnya. Kueri yang paling sederhana ialah mengambil seluruh elemen dari keals tersebut melalui method all(). Perhatikan field id yang secara otomatis diatur menjadi 1 dan 2 saat user tersebut ditambahkan.

Berikut ini cara lain melakukan kueri. Jika kita sudah mengetahui nilai id salah satu user, kita bisa mengambil objek user tersebut dengan perintah:

>>> u = User.query.get(1)

>>> u

<User john>

Sekarang tambahkan artikel baru:

>>> u = User.query.get(1)

>>> p = Post(body='my first post!', author=u)

>>> db.session.add(p)

>>> db.session.commit()

Penulis tidak memasukkan nilai untuk timestamp karena kita sudah mengaturnya secara otomatis di model. Lalu bagaimana dengan field user\_id? Perhatikan lagi bagain db.relationship yang ada di dalam kelas User dimana menambahkan sebuah atribut posts ke user, dan author ke posts. Penulis menambahkan seorang field virtual author (penulis) dan memasukkan objek user ke dalamnya langsung tanpa perlu menggunakan ID. SQLAlchemy sudah mengetahui apa yang harus dilakukan.

Untuk menyelesaikan bagian ini, mari kita lihat beberapa kueri lain:

>>> # get all posts written by a user

>>> u = User.query.get(1)

>>> u

<User john>

>>> posts = u.posts.all()

>>> posts

[<Post my first post!>]

>>> # same, but with a user that has no posts

>>> u = User.query.get(2)

>>> u

<User susan>

>>> u.posts.all()

[]

>>> # print post author and body for all posts

>>> posts = Post.query.all()

>>> for p in posts:

... print(p.id, p.author.username, p.body)

...

1 john my first post!

# get all users in reverse alphabetical order

>>> User.query.order\_by(User.username.desc()).all()

[<User susan>, <User john>]

Dokumentasi [Flask-SQLAlchemy](http://packages.python.org/Flask-SQLAlchemy/index.html) akan menjadi tempat terbaik untuk melihat semua opsi kuery database.

Mari kita harus seluruh data yang kita buat di atas sehingga database menjadi bersih dan siap digunakan di bab berikutnya:

>>> users = User.query.all()

>>> for u in users:

... db.session.delete(u)

...

>>> posts = Post.query.all()

>>> for p in posts:

... db.session.delete(p)

...

>>> db.session.commit()

## Shell Context

Masih ingat apa yang kita lakukan setelah memulai interpreter Python di bagian sebelumnya? Hal pertama yang kita lakukan adalah mengimpor modul dan keals yang diperlukan:

>>> from app import db

>>> from app.models import User, Post

Saat membuat aplikasi Flask, kita mungkin perlu menguji beberapa hal melalui shell Python sehingga mengulang-ulang impor di atas mungkin dapat membosankan. Perintah flask shell adalah opsi lain dari perintah flask yang akan memulai interpreter Python dan mengimpor seluruh kelas yang berhubungan dengan aplikasi Flask kita. Perhatikan contoh di bawah:

(venv) $ python

>>> app

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'app' is not defined

>>>

(venv) $ flask shell

>>> app

<Flask 'app'>

Saat memulai sesi interpreter biasa, app tidak dikenal karena belum diimpor, tapi saat menggunakan flask shell, app langsung dikenal karena sudah di impor secara otomatis. Kita juga bisa mengimpor data lain yang dibutuhkan melalui shell context.

Berikut ini fungsi yang ada di file microblog.py untuk membuat interpreter dari flask shell mengimpor db, User, dan Post secara otomatis:

from app import app, db

from app.models import User, Post

@app.shell\_context\_processor

def make\_shell\_context():

return {'db': db, 'User': User, 'Post': Post}

Decorator app.shell\_context\_processor mendaftarkan fungsi dibawahnya sebagai shell context function. Perintah flask shell selanjutnya akan mengimpor data yang di-return dan memberikan sebuah dictionary. Bagian yang ada di sebelah kiri tanda ":" adalah key, yang akan dipakai sebagai nama variabel saat diimpor.

Sekarang, saat memulai flask shell kita bisa langsung bekerja dengan database tanpa perlu mengimpor apapun:

(venv) $ flask shell

>>> db

<SQLAlchemy engine=sqlite:////Users/migu7781/Documents/dev/flask/microblog2/app.db>

>>> User

<class 'app.models.User'>

>>> Post

<class 'app.models.Post'>

<http://www.postgresqltutorial.com/postgresql-python/>

# How do I integrate a payment gateway in a website developed in PHP?

Ad by JetBrains

**[Develop PHP and Web projects with PhpStorm.](https://www.jetbrains.com/phpstorm/specials/phpstorm/phpstorm.html?utm_campaign=phpstorm&utm_medium=cpc" \t "_blank)**

[Great code assistance, smart debugger, safe refactorings, all major PHP frameworks support. Try for free!](https://www.jetbrains.com/phpstorm/specials/phpstorm/phpstorm.html?utm_campaign=phpstorm&utm_medium=cpc" \t "_blank)

[Download at jetbrains.com](https://www.jetbrains.com/phpstorm/specials/phpstorm/phpstorm.html?utm_campaign=phpstorm&utm_medium=cpc)

10 Answers

[Renish Aghera](https://www.quora.com/profile/Renish-Aghera-1)

[Renish Aghera](https://www.quora.com/profile/Renish-Aghera-1), 4+ years experience in PHP/MySQL, CodeIgniter, WordPress

[Answered Jun 17 2016](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Renish-Aghera-1)

If you want to integrate Paypal Payment gateway in your PHP website, take a look at some help on Paypal integration in PHP Website: [How to integrate payment system with PayPal in php & Mysql - iGreenTech Services](http://www.igreentechservices.com/how-to-integrate-payment-system-with-paypal-in-php-mysql/)

3.7k Views

### **Related Questions**[**More Answers Below**](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP#MoreAnswers)

* [How do I integrate payment gateway in PHP?](https://www.quora.com/How-do-I-integrate-payment-gateway-in-PHP)
* [How do I integrate a CommWeb payment gateway in PHP?](https://www.quora.com/How-do-I-integrate-a-CommWeb-payment-gateway-in-PHP)
* [How can I integrate payment gateway for my project website?](https://www.quora.com/How-can-I-integrate-payment-gateway-for-my-project-website)
* [What are the steps to develop one's own payment gateway in PHP?](https://www.quora.com/What-are-the-steps-to-develop-ones-own-payment-gateway-in-PHP)
* [How do I integrate bill desk payment gateway integration in PHP?](https://www.quora.com/How-do-I-integrate-bill-desk-payment-gateway-integration-in-PHP)

[**Ask New Question**](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP)

[Prashant Walke](https://www.quora.com/profile/Prashant-Walke)

[Prashant Walke](https://www.quora.com/profile/Prashant-Walke), Senior Software Engineer at Xento Systems (2018-present)

[Answered Apr 25](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Prashant-Walke) · Author has **59** answers and **5.9k** answer views

PayPal is the most popular payment gateway to send and receive payment worldwide. PayPal payment gateway is the easiest option for the web developer to implement payment system on the website.

Once the user clicks on the Buy Now/Payment button, they will be redirected to the PayPal where the payment will complete.  
After payment completion, the user will be redirected back to the website and the transaction details to be shown to the user.  
Also, the transaction information would be stored in the MySQL database.

PayPal has two environments such as

* Sandbox
* Live.

PayPal Sandbox allows developers to do their test transaction before the project go live.

**Create a sandbox account** from the below link :

[Create sandbox accounts](https://developer.paypal.com/docs/classic/lifecycle/sb_create-accounts/)

Live environment is used after project live. Once PayPal payment process is working fine on Sandbox environment, you can set PayPal to Live environment.  
**Database Tables Creation**

To store payment transaction information table need to be created in MySQL database. payments table is used for storing the transaction information provided by PayPal.

1. CREATE TABLE `payments` (
2. `payment\_id` **int**(11) NOT NULL AUTO\_INCREMENT,
3. `item\_number` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,
4. `txn\_id` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,
5. `payment\_gross` **float**(10,2) NOT NULL,
6. `currency\_code` varchar(5) COLLATE utf8\_unicode\_ci NOT NULL,
7. `payment\_status` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,
8. PRIMARY KEY (`payment\_id`)
9. )

**Submit a form with some predefined PayPal HTML** form field variable. Follow the comment tags (<!– –>) to know about the form hidden fields.

1. <?php
2. //Set useful variables for paypal form
3. $paypalURL = 'https://www.sandbox.paypal.com/cgi-bin/webscr'; //Test PayPal API URL
4. $paypalID = 'Insert\_PayPal\_Email'; //Business Email
5. ?>
6. <img src="images/<?php echo $row['image']; ?>"/>
7. Name: <?php echo $row['name']; ?>
8. Price: <?php echo $row['price']; ?>
9. <form action="<?php echo $paypalURL; ?>" method="post">
10. <!-- Identify your business so that you can collect the payments. -->
11. <input type="hidden" name="business" value="<?php echo $paypalID; ?>">
13. <!-- Specify a Buy Now button. -->
14. **<input** type="hidden" name="cmd" value="\_xclick"**>**
16. <!-- Specify details about the item that buyers will purchase. -->
17. **<input** type="hidden" name="item\_name" value="WP Database Backup"**>**
18. **<input** type="hidden" name="item\_number" value="12345"**>**
19. **<input** type="hidden" name="amount" value="22"**>**
20. **<input** type="hidden" name="currency\_code" value="USD"**>**
22. <!-- Specify URLs : replace with your site url -->
23. **<input** type='hidden' name='cancel\_return' value='http://walkeprashant.in/cancel.php'**>**
24. **<input** type='hidden' name='return' value='http://walkeprashant.in/success.php'**>**
26. <!-- Display the payment button. -->
27. **<input** type="image" name="submit" border="0"
28. src="button\_url" alt="PayPal - The safer, easier way to pay online"**>**
30. **</form>**

Change the cancel\_return and return input field value with your desired URL in PayPal HTML form.

**Payment Success** (success.php)

Once the PayPal payment is successful, the buyer is redirected to this page. The transaction information is received using $\_GET method and inserts payment data into the database.

Create success.php file and show appropriate message.

**Payment Cancel** (cancel.php)

If the buyer wishes to cancel payment at the PayPal payment page, the buyer is redirected to this page.

1. Your PayPal transaction has been canceled.

**IPN**

To make the PayPal Standard Payment more secure, Instant Payment Notification (IPN) should be used to validate the transaction.

Enable IPN  
To Use this feature, IPN must be enabled in PayPal account.  
Please go to the below link.  
[Identifying Your IPN Listener to PayPal](https://developer.paypal.com/docs/classic/ipn/integration-guide/IPNSetup/)

Add Notify URL in PayPal Form  
Add the following input field (notify\_url) HTML along with the other PayPal HTML Variables.

Validate Transaction  
Once IPN is enabled PayPal will send the transaction data to the Notify URL (<http://walkeprashant.in/ipn.php>). Place the following code in ipn.php file to validate and insert payment information into the database

1. **<php**
2. //Database credentials
3. $dbHost = 'localhost';
4. $dbUsername = 'root';
5. $dbPassword = '';
6. $dbName = 'walkeprashantdb';
8. //Connect with the database
9. $db = new mysqli($dbHost, $dbUsername, $dbPassword, $dbName);
11. //Display error if failed to connect
12. if ($db-**>**connect\_errno) {
13. printf("Connect failed: %s\n", $db->connect\_error);
14. exit();
15. }
16. $raw\_post\_data = file\_get\_contents('php://input');
17. $raw\_post\_array = explode('&', $raw\_post\_data);
18. $myPost = array();
19. foreach ($raw\_post\_array as $keyval) {
20. $keyval = explode ('=', $keyval);
21. if (count($keyval) == 2)
22. $myPost[$keyval[0]] = urldecode($keyval[1]);
23. }
25. // Read the post from PayPal system and add 'cmd'
26. $req = 'cmd=\_notify-validate';
27. if(function\_exists('get\_magic\_quotes\_gpc')) {
28. $get\_magic\_quotes\_exists = true;
29. }
30. foreach ($myPost as $key => $value) {
31. if($get\_magic\_quotes\_exists == true && get\_magic\_quotes\_gpc() == 1) {
32. $value = urlencode(stripslashes($value));
33. } else {
34. $value = urlencode($value);
35. }
36. $req .= "&$key=$value";
37. }
39. /\*
40. \* Post IPN data back to PayPal to validate the IPN data is genuine
41. \* Without this step anyone can fake IPN data
42. \*/
43. $paypalURL = "https://www.sandbox.paypal.com/cgi-bin/webscr";
44. $ch = curl\_init($paypalURL);
45. if ($ch == FALSE) {
46. return FALSE;
47. }
48. curl\_setopt($ch, CURLOPT\_HTTP\_VERSION, CURL\_HTTP\_VERSION\_1\_1);
49. curl\_setopt($ch, CURLOPT\_POST, 1);
50. curl\_setopt($ch, CURLOPT\_RETURNTRANSFER,1);
51. curl\_setopt($ch, CURLOPT\_POSTFIELDS, $req);
52. curl\_setopt($ch, CURLOPT\_SSLVERSION, 6);
53. curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, 1);
54. curl\_setopt($ch, CURLOPT\_SSL\_VERIFYHOST, 2);
55. curl\_setopt($ch, CURLOPT\_FORBID\_REUSE, 1);
57. // Set TCP timeout to 30 seconds
58. curl\_setopt($ch, CURLOPT\_CONNECTTIMEOUT, 30);
59. curl\_setopt($ch, CURLOPT\_HTTPHEADER, array('Connection: Close', 'User-Agent: company-name'));
60. $res = curl\_exec($ch);
62. /\*
63. \* Inspect IPN validation result and act accordingly
64. \* Split response headers and payload, a better way for strcmp
65. \*/
66. $tokens = explode("\r\n\r\n", trim($res));
67. $res = trim(end($tokens));
68. if (strcmp($res, "VERIFIED") == 0 || strcasecmp($res, "VERIFIED") == 0) {
70. //Payment data
71. $item\_number = $\_POST['item\_number'];
72. $txn\_id = $\_POST['txn\_id'];
73. $payment\_gross = $\_POST['mc\_gross'];
74. $currency\_code = $\_POST['mc\_currency'];
75. $payment\_status = $\_POST['payment\_status'];
77. //Check if payment data exists with the same TXN ID.
78. $prevPayment = $db->query("SELECT payment\_id FROM payments WHERE txn\_id = '".$txn\_id."'");
79. if($prevPayment->num\_rows > 0){
80. exit();
81. }else{
82. //Insert tansaction data into the database
83. $insert = $db->query("INSERT INTO payments(item\_number,txn\_id,payment\_gross,currency\_code,payment\_status) VALUES('".$item\_number."','".$txn\_id."','".$payment\_gross."','".$currency\_code."','".$payment\_status."')");
84. }
86. }

**Make PayPal Payment Gateway Live**

When your application payment flow testing is completed, you need to modify two files to make PayPal payment gateway live.

979 Views

[Promoted by DigitalOcean](https://try.digitalocean.com/performance/?utm_medium=paid_social&utm_campaign=Quora_Display%7CPricePerf%7CWebAppTopics" \t "_blank)

**[Build, test and deploy something new. Free for 60 days.](https://try.digitalocean.com/performance/?utm_medium=paid_social&utm_campaign=Quora_Display%7CPricePerf%7CWebAppTopics" \t "_blank)**

[Save up to 25% on your CI, data analysis and batch processing nodes. Try free for 60 days.](https://try.digitalocean.com/performance/?utm_medium=paid_social&utm_campaign=Quora_Display%7CPricePerf%7CWebAppTopics" \t "_blank)

[Learn more at try.digitalocean.com](https://try.digitalocean.com/performance/?utm_medium=paid_social&utm_campaign=Quora_Display%7CPricePerf%7CWebAppTopics)

[Harsh Gaur](https://www.quora.com/profile/Harsh-Gaur)

[Harsh Gaur](https://www.quora.com/profile/Harsh-Gaur), PHP Developer at Walkover Web Solutions (2018-present)

[Answered Feb 7 2017](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Harsh-Gaur) · Author has **63** answers and **15.4k** answer views

Hi There,

Each and every payment Gateway provider provides sets of API with documentation to integrate in multiple language. To be more specific you need to be sure which payment gateway you will be using.

For example, one my favorite payment gateway is [Razorpay · Payment Gateway Solution for India](https://razorpay.com/) .

Now if you want to use this payment gateway you will find complete documentation at [Getting Started with Razorpay · Razorpay](https://docs.razorpay.com/v1/docs) .

Before integrating you need to register and generate set you access ids. Once you are done you can use [PHP: cURL - Manual](http://php.net/manual/en/book.curl.php) to make the calls to their APIs.

If you have any other doubts do reach me again.

1.6k Views

[Promoted by MongoDB](https://www.mongodb.com/cloud/atlas/lp/general?utm_medium=PS&utm_campaign=WW_Atlas_Reg_D_1.11.18&utm_content=i-1&utm_term=3&jmp=quora" \t "_blank)

**[MongoDB Atlas: Built for your most sensitive workloads.](https://www.mongodb.com/cloud/atlas/lp/general?utm_medium=PS&utm_campaign=WW_Atlas_Reg_D_1.11.18&utm_content=i-1&utm_term=3&jmp=quora" \t "_blank)**

[Automated MongoDB service built for the most sensitive workloads at any scale. Get started free.](https://www.mongodb.com/cloud/atlas/lp/general?utm_medium=PS&utm_campaign=WW_Atlas_Reg_D_1.11.18&utm_content=i-1&utm_term=3&jmp=quora" \t "_blank)

[Free trial at mongodb.com](https://www.mongodb.com/cloud/atlas/lp/general?utm_medium=PS&utm_campaign=WW_Atlas_Reg_D_1.11.18&utm_content=i-1&utm_term=3&jmp=quora)

[Saurabh Suman](https://www.quora.com/profile/Saurabh-Suman-12)

[Saurabh Suman](https://www.quora.com/profile/Saurabh-Suman-12), CEO at www.Paydeck.in

[Answered Feb 8 2017](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Saurabh-Suman-12) · Author has **1.8k** answers and **2.6m** answer views

While I was working at CanDeotech for approx 12 years, I used to get a lot of website projects. And almost 7 out of 10 was to add a PG into them. We used to work mainly on LAMP stack. Those were mostly overseas projects and PG = Paypal was the common ask point.

So we had a script ready for that which we used to simply integrate. Now there are many API’s & SDK’s available to do so for PHP. I am talking about 2004–2013.

Post 2013, we started working on Indian websites too. And the best thing which I later found was JUSPAY.

JUSPAY is a payment wrapper. So if you will integrate one JUSPAY wrapper into a website, you wont have to again and again work on script integration or wont have to spend money on development hours.

JUSPAY covered India’s almost all Big PG’s.

And the process is quite simple!

Register on JusPay.

Integrate the JUSPAY console for once. Get through with testing & results.

Later whenever & how many PG’s available in India you signup with, you can simply let those PG companies know, that you have JUSPAY as a wrapper. They will provide you with JUSPAY clearance at their backend & their Credentials.

You dont need to be a developer to then add any pg, whether its PAYU, RAZORPAY, EBS, CCAVENUE, etc.

Once you get your application approved with the PG company, you will receive the credentials in your Registered Email address.

Login to your JUSPAY account, Select your PG & add those credentials, like you fill a form, and your PG is live!

Adding more awesomeness is, if you have 3 PG’s, JUSPAY gives your Priority LOGIC setup, which has basic & advanced setting options. And you can use all 3 PG’s at your PHP website.

I will advise you to surely try JUSPAY, if you are a developer. It will save you lot of time & money!

PS: I dont work for JUSPAY. Just that I understand the PAIN the developer goes through everytime while integrating PAYMENT Gateways.

Thanks.

Saurabh - Working at [Pay with Credit Card anytime, anywhere & anyone | PAYDECK](http://paydeck.in/)

1.4k Views

[Shivani Ankush Sharma](https://www.quora.com/profile/Shivani-Ankush-Sharma)

[Shivani Ankush Sharma](https://www.quora.com/profile/Shivani-Ankush-Sharma), Owns a Startup!

[Answered Dec 4 2015](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Shivani-Ankush-Sharma) · Author has **522** answers and **292.2k** answer views

To integrate a Payment Gateway into your website, you must always hire a professional and experienced company to ensure security and reliability, facilitating safe online transactions.  
Since a Payment Gateway is the most important aspect of any online business, a professionally qualified company seamlessly integrates any Payment Gateway of your choice, making shopping carts friendly, secure and reliable, to business owners and their customers.  
There's a company that I know of which is helping Online Businesses with online transactions. The company is **Enuke Software Pvt. Ltd.** To knw about their work, you can visit [Payment Gateway Integration](http://www.enukesoftware.com/payment-gateway-integration-service-company.html)

1.4k Views

[Abhishek Banerjee](https://www.quora.com/profile/Abhishek-Banerjee-5)

[Abhishek Banerjee](https://www.quora.com/profile/Abhishek-Banerjee-5), Software Engineer

[Answered Feb 12 2015](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Abhishek-Banerjee-5) · Author has **53** answers and **51.1k** answer views

Check out Stripe [Documentation](https://stripe.com/docs).  
Or take a look at EBS.  
Links:  
[Payment Gateway India, Merchant Account and Credit Card Processing](https://www.ebs.in/en/)  
[https://support.ebs.in/app/index...](https://support.ebs.in/app/index.php?%2Fdefault_import%2FKnowledgebase%2FList%2FIndex%2F7%2Fshopping-cart-kits)

2.9k Views · [View Upvoters](https://www.quora.com/api/mobile_expanded_voter_list?key=RESJfZ1cbzK&type=answer) · Answer requested by [Abadhesh Savita](https://www.quora.com/profile/Abadhesh-Savita)

[Aleksandra Seremina](https://www.quora.com/profile/Aleksandra-Seremina)

[Aleksandra Seremina](https://www.quora.com/profile/Aleksandra-Seremina)

[Answered May 16, 2016](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Aleksandra-Seremina) · Author has **157** answers and **155.4k** answer views

The simplest way is to use a Paypal button adding it to the appropriate place at your website. However, there is another more complex but more efficient way. You can create your own HTML-form that contains all the necessary data for making payments.

There are ready-made modules for a majority of CMS and they allow integrating Paypal payment gateway with an e-commerce site. For example, a module for Drupal 7 and his module Commerce: [Commerce PayPal | Drupal.org](https://www.drupal.org/project/commerce_paypal)

If you interested in some specific details and tips on Paypal integration, take a look here: [http://cases.azoft.com/integrate...](http://cases.azoft.com/integrate-paypal-payment-system-website/)

1.6k Views · [View Upvoters](https://www.quora.com/api/mobile_expanded_voter_list?key=40DWyRg3OCj&type=answer)

[Harish Kumar](https://www.quora.com/profile/Harish-Kumar-1729)

[Harish Kumar](https://www.quora.com/profile/Harish-Kumar-1729), Web Application Development

[Answered Sep 9, 2016](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Harish-Kumar-1729) · Author has **347** answers and **73.1k** answer views

You can contact [**iPistis Technologies pvt. ltd.**](https://www.ipistis.com/) , this company is providing integration services for almost all payment gateway in PHP

[**Payment Gateway Integration - iPistis Technologies Pvt. Ltd.**](https://www.ipistis.com/payment-gateway-integration/)

436 Views

[Sudeep Mukherjee](https://www.quora.com/profile/Sudeep-Mukherjee-3)

[Sudeep Mukherjee](https://www.quora.com/profile/Sudeep-Mukherjee-3), Achieving new horizon

[Answered Jan 14, 2015](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Sudeep-Mukherjee-3) · Author has **573** answers and **1.3m** answer views

For Reference you can use [Card Payments for Membership with PHP](https://www.youtube.com/playlist?list=PLfdtiltiRHWHn4uErhCQEk1EIUjSGkuDk). You can also use other api based on price and scalability

1.3k Views

[Deepak Kapoor](https://www.quora.com/profile/Deepak-Kapoor-1)

[Deepak Kapoor](https://www.quora.com/profile/Deepak-Kapoor-1), Developer. Database manager.

[Answered Feb 13, 2015](https://www.quora.com/How-do-I-integrate-a-payment-gateway-in-a-website-developed-in-PHP/answer/Deepak-Kapoor-1) · Author has **66** answers and **106.3k** answer views

Each payment gateway has its own api which asks you to send some basic info to a url like  
Initial page  
Order number amount  
Failure page  
Success page