Creating Your Database

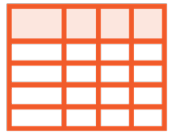
Xavier Morera

HELPING DEVELOPERS UNDERSTAND SEARCH & BIG DATA

@xmorera www.xaviermorera.com

Creating Databases with SQLAlchemy

Create databases



Create tables

Constraints and default values

Insert data

Create a Database Using SQLAlchemy

Create using a command

Created automatically

```
engine = db.\
create_engine('mysql+mysqlconnector://root:mysql@localhost:3306/')

connection = engine.connect()

engine.execute('SHOW DATABASES;').fetchall()

engine.execute('CREATE DATABASE test_mysql_sa;')
```

Creating MySQL Databases

Create database using a SQL command

Please note connection string: No database specified

Execute CREATE DATABASE statement

```
engine = db.create_engine('sqlite:///new_sqlite.db')

connection = engine.connect()
```

Creating SQLite Databases

Created automatically

Specify database name in connection string

- Careful of typos

```
engine =
db.create_engine('mysql+mysqlconnector://root:mysql@localho
st:3306/test_mysql_sa')

connection = engine.connect()

metadata = db.MetaData()
```

Creating Tables

Start with the usual create_engine and connect

Create database MetaData

- Container object

```
mysql> USE test_mysql_sa;
Database changed
mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql>
```

```
posts = db.Table('posts', metadata,
    db.Column('Id', db.Integer()),
    db.Column('Title', db.String(255)),
    db.Column('ViewCount', db.Integer()),
    db.Column('Question', db.Boolean()))
metadata.create_all(engine)
```

Creating Tables with Table

Define Table object with name and metadata

- Each Column, with type

Create using create_all

- Use table.create for single table

```
posts

list(posts.c)

dir(posts.c)
```

Creating Tables with Table

Created a Table object

- Columns

Inspect our object

```
mysql> DESCRIBE posts;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| Id        | int(11)      | YES  |     | NULL    |       |
| Title     | varchar(255) | YES  |     | NULL    |       |
| ViewCount | int(11)      | YES  |     | NULL    |       |
| Question  | tinyint(1)   | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

```
mysql> SELECT * FROM posts;
mysql> INSERT INTO posts (Id, Title) VALUES (0, 'Is Data Science the Future?');
mysql> SELECT * FROM posts;
mysql> INSERT INTO posts (Id, Title) VALUES (0, 'Is Data Science the Future?');
mysql> SELECT * FROM posts;
mysql> INSERT INTO posts (Title) VALUES ('Is Data Science the Future?');
mysql> SELECT * FROM posts;
```

Why More Than Just Columns

The problem

- Test inserting repeated and missing values

A Few Problems Here

```
posts_two = db.Table('posts_two', metadata,
  db.Column('Id', db.Integer(), primary_key=True, unique=True),
  db.Column('Title', db.String(255), nullable=False),
  db.Column('ViewCount', db.Integer(), default=1000),
  db.Column('Question', db.Boolean(), default=True))

posts_two.create(engine)
```

Constraints and Default Values

Create Table object

Use: primary_key, unique, nullable, and default

```
mysql> describe posts_two
mysql> INSERT INTO posts_two (Id, Title) VALUES (0, 'Is Data Science the Future?’);
mysql> INSERT INTO posts_two (Id, Title) VALUES (0, 'Is Data Science the Future?');
mysql> SELECT * FROM posts;
```

Constraints and Default Values

Use describe to inspect table

Insert values from MySQL

- Not the ORM

```
engine = db.create_engine('sqlite:///sqlalchemy_sqlite.db', echo=True)
connection = engine.connect()
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship
Base = declarative_base()

class User(Base):
  __tablename__ = 'user'
  Id = db.Column(db.Integer(), primary_key=True)
  Name = db.Column(db.String())
```

Declarative Class Definitions with Constraints

Use the Declarative API

- With declarative_base

Create a class

- Define fields

```
class Post(Base):
  __tablename__ = 'post'
  Id = db.Column(db.Integer(), primary_key=True)
  Title = db.Column(db.String(255), nullable=False)
  ViewCount = db.Column(db.Integer(), default=1000)
  Question = db.Column(db.Boolean(), default=True)
  OwnerUserId = db.Column(db.Integer(), db.schema.ForeignKey('user.Id'),
nullable=False)
  User = relationship('User', backref='post')


Base.metadata.create_all(engine)
```

Declarative Class Definitions with Constraints

Create a second class

Define the ForeignKey and relationship

Create and review in sqlite

```
$ SQLite3 sqlalchemy_sqlite.db
sqlite> .tables
sqlite> .schema post
```

Declarative Class Definitions with Constraints

Confirm using SQLite3

- List tables
- Inspect schema

```
engine = db.create_engine('sqlite:///sqlalchemy_sqlite.db')
connection = engine.connect()
metadata = db.MetaData()

users = db.Table('user', metadata, autoload=True, autoload_with=engine)
stmt = db.insert(users).values(Name='Xavier Morera')
connection_result = connection.execute(stmt)
```

Insert Using a Statement

Create a Table object

Use insert with the table

- Include values

Execute statement and confirm

```
from sqlalchemy.orm import sessionmaker

session = sessionmaker()

session.configure(bind=engine)

my_session = session()
```

Insert Using Session

Import sessionmaker

Configure Session class, associating it with our engine

Work with the session

```
Juli = User(Name='Juli')
Luci = User(Name='Luci')
my_session.add(Juli)
my_session.add(Luci)
session.new
my_session.commit()
for each_user in my_session.query(User).all():
    print(each_user.Name)
```

Insert Using Session

Create instances of our User class

Then add to the session and commit

Confirm

```
posts = db.Table('post', metadata, autoload=True,
autoload_with=engine)

stmt = db.insert(posts)

values_list = [{'Title': 'Data Science Question', 'OwnerUserId':1},

  {'Title': 'Data Science Answer', 'OwnerUserId':1]

result = connection.execute(stmt, values_list)
```

Insert Multiple Records

Insert using a statement

- Specify the table, prepare/pass the values
- And use insert

```
one_post = Post(Title='Sample question', OwnerUserId=1)
one_answer = Post(Title='Sample answer', Question=False,
OwnerUserId=1)
my_session.add_all([one_post, one_answer])
my_session.new
my_session.commit()
```

Insert Multiple Records

Insert using session

- With add_all
- Passing an iterable

Observe the default value

```
engine = db.create_engine('sqlite:///sqlalchemy_csv.db')
connection = engine.connect()
with open('tags.csv', 'r') as file:
    tags_df = pd.read_csv(file)
tags_df.head()
tags_df.to_sql('tags', con=engine)
engine.execute('select * from tags limit 10;').fetchall()
```

Loading a CSV into a Table

Load the CSV into a pandas DataFrame

- Using read_csv

Write to database using to_sql

- Confirm

Takeaway

Create databases

- Using a command
- Automatically

Create tables

- Table object
- Multiple Column objects
- Constraints and default values

Takeaway

Insert Data

- Single and multiple rows
- Statement or using session

Load CSV into a table

- Pandas + SQLAlchemy
- Applies to other file formats