SQLAlchemy

Eduardo Mendes

Roteiro - Parte 1

Introdução

- O que é o SQLAlchemy
- Por que usar o SQLAlchemy
- SQLAlchemy Core
- SQLAlchemy ORM
- Como escolher entre o Core e o ORM

Roteiro - Parte 2

SQLAlchemy Core

- Esquemas e tipos
  - Tipos e Metadados
  - tabelas e colunas
  - Chaves e restrições
  - Indexes
  - Relações
- Inserindo dados
- Buscando dados
- Atualizando dados
- Deletando dados

Roteiro - Parte 3

SQLAlchemy ORM

- Como funciona um ORM?
- Esquemas e tipos
  - Tipos e Metadados
  - tabelas e colunas
  - Chaves e restrições
  - Indexes
  - Relações
- Inserindo dados
- Buscando dados
- Atualizando dados
- Deletando dados

Parte 1 - Intro

O que é o SQLAlchemy

SQLAlchemy é uma biblioteca, criada por Mike Bayer em 2005, usada para interagir com uma grande variedade de bancos de dados.

Ele permite que você crie modelos de dados e consultas de uma maneira que se sente como classes e declarações normais de Python.

Por que usar SQLAlchemy?

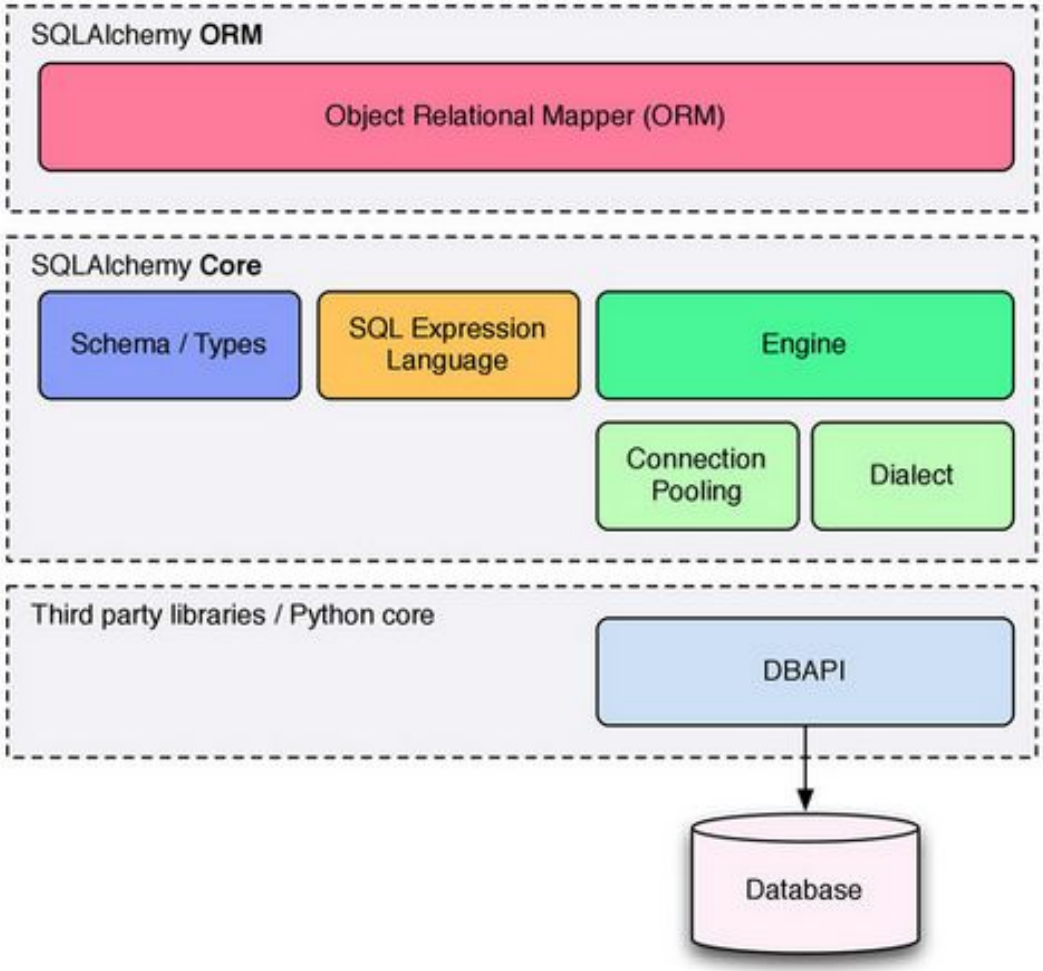
- Abstrair seu código de SQL
- Aproveitar declarações e tipos comuns instruções SQL sejam criadas de forma eficiente
- Evitar problemas comuns, como ataques de injeção SQL
- É um banco extensível
    - Além de trabalhar com Oracle, Mysql, postgres e etc.. Pode ser estendida facilmente para outros bancos relacionais
- Trabalha em dois modos diferentes (Core e ORM)

SQLAlchemy Core

- Uma maneira Pythonica de representar seus dados sem perder o sotaque do SQL
- Focado diretamente no banco e seu esquema
- Proporciona a base para o ORM

SQLAlchemy ORM

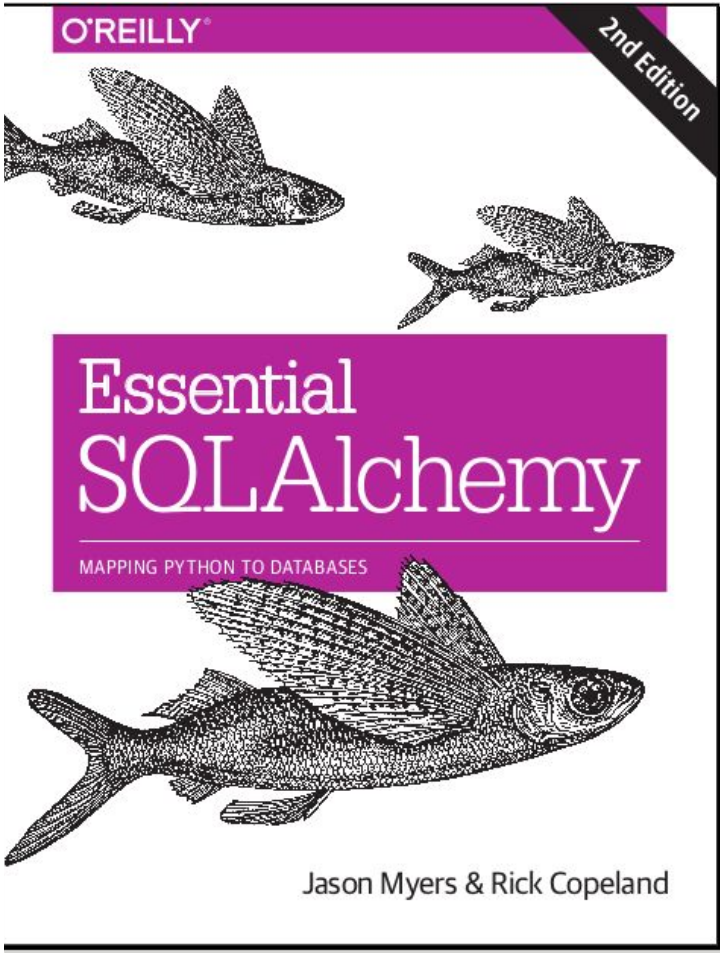
- Se parece tradicionalmente com outros ORMs
- Fornece uma grande camada de abstração do SQL
  - Porém converte tudo para a camada do core
- Trata tabelas e dados com a abstração de classes e objetos
- Pode ser usada com o core para camadas mais abstratas



https://speakerdeck.com/zzzeek/introduction-to-sqlalchemy-and-orms-pgopen-2013

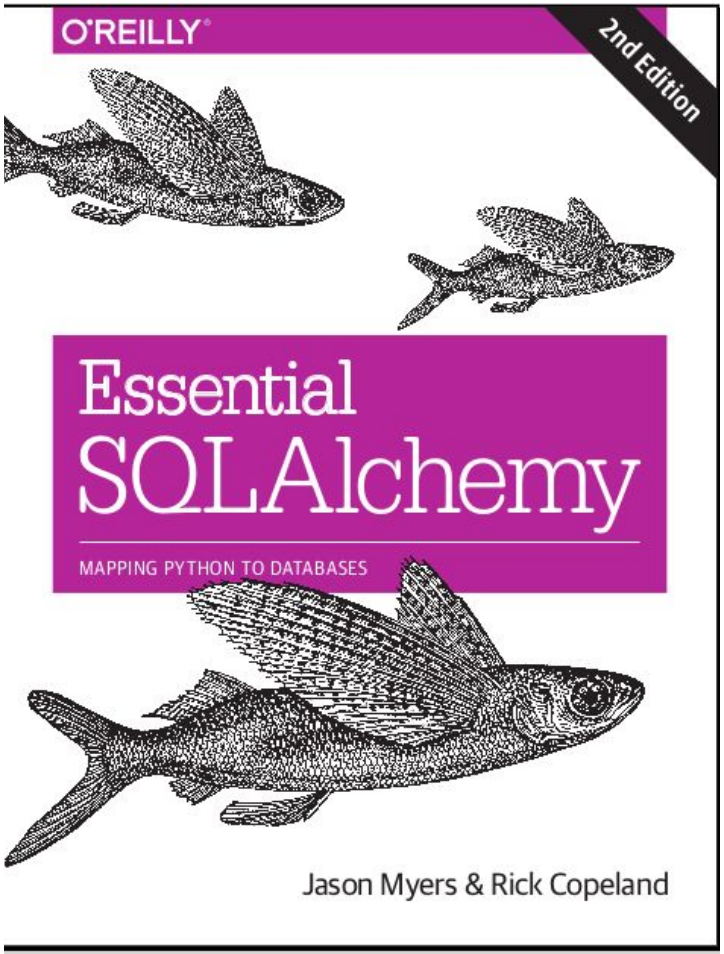
Qual usar? Core x ORM [0]

- Se você já trabalha com ORM, mas quer adicionar mais funcionalidades, use o **core**
- Se você quer ver seus dados no formato do esquema SQL, use o **core**
- Se você não tem uma camada bem definida de negócios, use o **core**
- Caso tenha regras de negócios bem definidas, use o **ORM**



Qual usar? Core x ORM [1]

- Se você está construindo um protótipo rápido, use o **ORM**
- Se você tem uma combinação de necessidades, use **AMBOS**



Parte 2 - Core

Tipos ->

- Metadados
  - Facilita a indexação e busca na tabela
- Table
  - De fato a table
- Colunas
  - De fato colunas
- Index
  - Acelera a busca em um field expecífico

SQLAlchemy	Python	SQL
BigInteger	int	BIGINT
Boolean	bool	BOOLEAN or SMALLINT
Date	datetime.date	DATE (SQLite: STRING)
DateTime	datetime.datetime	DATETIME (SQLite: STRING)
Enum	str	ENUM or VARCHAR
Float	float or Decimal	FLOAT or REAL
Integer	int	INTEGER
Interval	datetime.timedelta	INTERVAL or DATE from epoch
LargeBinary	byte	BLOB or BYTEA
Numeric	decimal.Decimal	NUMERIC or DECIMAL
Unicode	unicode	UNICODE or VARCHAR
Text	str	CLOB or TEXT
Time	datetime.time	DATETIME

Exemplo de código

```
users_table = Table('usuarios', metadata,
                    Column('id', Integer, primary_key=True),
                    Column('nome', String(40), index=True),
                    Column('idade', Integer, nullable=False),
                    Column('senha', String),
                    Column('criado_em', DateTime(), default=datetime.now),
                    Column('atualizado_em',
                           DateTime(),
                           default=datetime.now,
                           onupdate=datetime.now))
```

CODE!!!